DOCUMENT RESUME

ED 126 855 IR 003 759

AUTHOR Follettie, Joseph F., Ed.; Teplitzky, Frank, Ed.
TITLE Computer-Based Systems to Facilitate Instructional Development.
INSTITUTION Southwest Regional Laboratory for Educational Research and Development, Los Alamitos, Calif.
PUB DATE 72
NOTE 178p.; SWRL Working Papers: 1972

EDRS PRICE MF-$0.83 HC-$10.03 Plus Postage.
DESCRIPTORS Computer Oriented Programs; *Computer Programs; Documentation; *Educational Development; *Educational Research
IDENTIFIERS Language Analysis Package; *Southwest Regional Laboratory; SWRL

ABSTRACT

This volume reproduces a sample of Southwest Regional Laboratory for Educational Research and Development (SWRL) Technical Notes and Technical Memoranda produced in 1972 dealing with natural language analysis and student-instruction interaction facets of the organization's computer-based research effort. Part I contains three papers describing aspects of the SWRL Language Analysis Package. Part II contains four papers dealing with the Instructional Development Control and Monitoring System. Part III presents abstracts of SWRL Technical Notes and Technical Memoranda produced in 1971 and 1972 bearing on natural language analysis, student-instruction interaction, and related matters. (HAB)

COMPUTER-BASED SYSTEMS TO FACILITATE INSTRUCTIONAL DEVELOPMENT

Edited by Joseph F. Follettie and Frank Teplitzky

SWRL EDUCATIONAL RESEARCH AND DEVELOPMENT
4665 Lampson Avenue, Los Alamitos, California

2

PREFACE

The documentation of large-scale development endeavors in education is a phenomenon with which the educational R&D community has had modest experience, since there has been little large-scale development to document. SWRL documentation experience confirms the applicability of Derek Price's conclusion regarding the literature of research and the literature of development.

> A scholarly publication is not a piece of information but an expression of the state of a scholar or a group of scholars of a particular time. We do not, contrary to superstition, publish a fact, a theory, or a finding, but some complex of these . . . . If the paper is an expression of a person or several persons working at the research front, we can tell something about the relations among the people from the papers themselves . . . . It seems that technologists differ markedly from both scientific and nonscientific scholars. They have a quite different scheme of social relationships, are differently motivated, and display different personality traits [Price, 1970, pp. 7-9].

> Clearly, the published paper is not, in general, the end product of a worker in a technological subject; he appears to be instead concerned chiefly with the production of an artifact or process. What then is the role of literature in technology? I suggest that for the most part it is produced as an epiphenomenon. It comes about because many technologists have had scientific training and know full well the code of behavior of the scientist in which publication is not merely right and proper, but a high duty and a behavior expected by peers and employers . . . . In general new technology will flow from old technology rather than from any interaction there might be between the analogous but separate structures of science and technology [Price, 1965, pp. 560-561].

SWRL experience has been that the course of a well-managed development effort produces considerable documentation but that a good deal of the substance of the information exceeds structures and strictures of journal publication. The journal article constitutes an available medium, but the laundering of the information required to use the medium often washes out the message.

SWRL has found it unproductive to treat information and documentation in the abstract as a "communication problem." A more useful approach is to consider operational means of making information pertinent to large-scale development in education conveniently available to interested audiences. This perspective directs attention to specifying interested audiences and devising communication compatible with their need-to-know characteristics. SWRL information architecture recognizes several audiences.

Staff involved in the development per se and the contract sponsor are two of the most immediate audiences addressed by SWRL documentation. Communication relevant to these audiences is handled by SWRL Technical Notes and Technical Memoranda that chronicle the course of SWRL R&D. These documents range in length from a few to a few hundred pages depending upon their nature. Some 200 of these Technical Notes and Technical Memoranda are issued during the course of a year--a stack several feet tall.

A third audience is the invisible colleges in which SWRL staff actively participate. Collegial exchange of selected Technical Notes and Technical Memoranda serve this audience adequately.

Another audience is product users. A volume of product working papers that brings together the documents associated with the development of each product is issued at the time the product is made available for general use and provides relevant information for this audience.

This leaves the general audience of students, scholars, and other members of the R&D community in education. SWRL Technical Reports and Professional Papers, largely accessed via the ERIC system, are directed to this broad audience. Journals, professional meetings, and other classical scientific and technical information exchange mechanisms are also used.

But each of these mechanisms involves a packing and rationalizing of information into independent pieces that inherently involves time delays and loses some of the original flavor of the work in the process. To reduce the time interval and retain the freshness of the work, an Annual Working Papers series has been initiated. The thematic topics that provide convenience categories for representing inquiry completed during the past year that is of timely interest to a sector of the educational R&D community will be identified. The documents relevant to these topics will then be organized into the volumes constituting

the Annual Working Papers for that year. "Computer-Based Systems to Facilitate Instructional Development" is one of four such volumes for the year 1972. The other three volumes of the 1972 SWRL Working Papers, available through the ERIC system, include the following titles:

Design of an Instructional Management System
(John F. McManus, editor)

The Integration of Content, Task, and Skills Analysis Techniques in Instructional Design (David W. Bessemer and Edward L. Smith, editors)

Prototype Testing in Instructional Development
(Fred C. Niedermeyer, editor)

CONTENTS

COMPUTER-BASED SYSTEMS TO FACILITATE INSTRUCTIONAL DEVELOPMENT

Joseph F. Follettie and Frank Teplitzky

The use of computer-based systems as functional tools to facilitate R&D has received a good deal of attention in fields other than education. The effort directed to this end has had highly beneficial effects on the R&D in these fields. In education, the interest in the computer as an instructional device has so occupied attention that potentials for computer-based systems as utility/tools in educational R&D per se has all but escaped notice.

Acquiring and using state-of-the-art systems developed to date in other fields is a routine necessity for a large-scale R&D organization in education. After doing this, SWRL has encountered still additional unique requirements to support the systematic development of instruction. These requirements are not satisfied by current shelf items. This document pertains to two such requirements to which SWRL effort has been directed.

The volume reproduces a sample of SWRL Technical Notes and Technical Memoranda produced in 1972 dealing with natural language analysis and student-instruction interaction facets of the organization's computer-based research effort. The natural language analysis activity centers on development of a Language Analysis Package (L.A.P.). Analogous to the UCLA Bi-Med statistical analysis programs, the interactive modularized L.A.P. permits a user to analyze natural language samples in various ways while using any of a variety of remote computers. The

student-instruction interaction activity centers on design, development, and extension of an Instructional Development Control and Monitoring System (IDCMS), a multiple-station instructional research system whose controller is a small computer.

Part I contains three papers describing aspects of the Language Analysis Package. Part II contains four papers dealing with the Instructional Development Control and Monitoring System. Part III presents abstracts of SWRL Technical Notes and Technical Memoranda produced in 1971 and 1972 bearing on natural language analysis, student-instruction interaction, and related matters.

Working Paper 1

LANGUAGE ANALYSIS PACKAGE (L.A.P.) VERSION I SYSTEM DESIGN (TM 5-72-06)

Ann Porch

1.0.0 INTRODUCTORY CONCEPTS

For years there have been "packaged programs" in statistical areas. These programs offer generalized computational capabilities in a form and format especially suited to easy use by researchers whose basic orientation is not that of Computer Science.[1] Researchers in the social sciences, for instance, are able to perform complex multi-variate regression analysis by computer without undergoing any special training in programming or computer operations.

A Language Analysis Package (L.A.P.) with a power comparable to that of statistical packages will have considerable general utility.

It will permit researchers to use the speed and versatility of the computer to process natural language text as well as numerical data. For example, researchers doing studies of textbooks typically analyze their data by hand. Where computer technology is employed, a special-purpose program is generally written by the resident programmer, who may not have specialized training in techniques of natural language processing. The results are costly, both in time and money spent on processing with inefficient or one-shot programs. Because such programs are limited in scope and written for a special purpose, the researcher finds that a relatively minor change in his research perspective makes the computer program unusable.

---

[1] See Dixon, W. J. (ed.), Biomedical Computer Programs, Univ. of Calif. Press, (1970) and Nie, N. et al. (ed.), Statistical Package for the Social Sciences, McGraw Hill, (1970).

During the past ten years, a great deal of work has been done in natural language processing throughout the world in fields such as artificial intelligence, information retrieval, machine translation, computational linguistics, and computer stylistics. Hundreds of computer programs have been written, debugged, run, and then shelved when the researcher went on to another project. A number of these programs are the product of months of careful work by experts.

1.1.0 Rationale for Package Development

The design proposed here suggests the use of the best of those existing programs whose authors are willing to release them, together with programs written specially for the package. Such an approach has several advantages: the package will reflect the power of the finest specialized programming skill presently available; the development costs will be minimized, since one major programming task will consist of interfacing the existing programs or subsections in a modular fashion under the direction of one control routine, rather than developing each of the specialized routines from scratch. By carefully constructing the package in a highly independent, modular manner, individual routines may be easily "un-plugged" and replaced when a more efficient or powerful routine is developed or should a new approach indicate combinations of functions. Thus, the system will be dynamic and open-ended, capable of being easily updated to keep pace with the state-of-the-art.

The package will be developed in several stages. This document primarily describes the limited capability of Version I, but will often refer to more powerful capabilities to be incorporated in later versions.

Care will be taken to "tag" these future features and to differentiate them from the system design for Version I.

All development will try to keep the entire package as machine independent as possible. Certainly it will be implemented on a computer which can compile and run most of the major computer languages currently in use for language processing. One such installation exists at UCLA, where the IBM 360 mod 91 has compilers for the following languages: PL/1, FORTRAN IV (G & H), COBOL, SNOBOL, LISP, APL, 360 ASSEMBLER, and ALGOL. Its operating system also allows for interfacing subroutines written in different source languages.

Certain hardware and system software requirements will be essential to the efficient development of the L.A.P. Among these are an efficient system sort-merge routine, multiple tape drives, relatively large amounts of direct-access storage and considerable available core. Again, the UCLA installation is one example of a computer center which is amply equipped in all areas.

Subsequent versions of the Language Analysis Package will have the ability to perform efficiently in all basic areas of natural language analysis, and the flexibility to operate either on-line or in batch mode. In addition, beginning with Version I, the L.A.P. will be easily modifiable at any time, either for more effective general use or for a particular research application.

Flexibility of use will be a major consideration in the development of such a package. Version I of the L.A.P. will provide the user with many options, allowing him to select precisely and easily only the functions he requires. No section of the package will be

"called-in" unless the researcher specifically requests it. He will not be limited to simply an exclusive "OR" type selection, where he can only chose to do either a KWIC or an Index, but will be able to combine routines and subroutines in the logical order he desires. He may, for example, want to produce KWIC's on words occurring within his inclusion list while simultaneously producing an index of all words except those in his exclusion list and a frequency count of every word in the text. He will be able to use only the retrieval aspect of the L.A.P. or only the statistical portion without being penalized by the fact he is using a package rather than a single program designed specifically for his purpose.

1.2.0. Types of Processing for Natural Language

A survey of the work currently being done in the field of language analysis[2] reveals seven major areas of present interest and usage which can logically be included in Version I of the L.A.P. Of the seventy-five projects listed in the November, 1971 issue of Computers and the Humanities, nine dealt with frequency counts, seventeen with KWIC production, fourteen with semantic or content analysis (including automatic abstracting), eight with statistics, thirteen with index production, six with retrieval systems, six with sentence parsing and six with miscellaneous items such as machine translation. Several projects must be considered to fall into more than one category. Ratios in Linguistics in Documentation (Current

[2] See Porch, Ann, "People and Projects in Natural Language Processing: A Preliminary Bibliographic Directory" TM 5-71-10, August 5, 1971, 107 pages.

Abstracts), Language and Automation, and Computer Studies in the Humanities and Verbal Behavior are much the same, although with a slightly heavier emphasis on retrieval and parsing.

1.3.0 Data Bases and Their Manipulation

Since a number of researchers will be using the L.A.P., Version I of the system will have the ability to differentiate among data bases, selecting the researcher's base or sub-base from a library of resident data bases stored on magnetic tape or disc and making it available for his processing. In later versions, the entire library may include the ERIC files, or state adopted textbooks. Researchers using the L.A.P. will be able to obtain an input file containing only first grade reading books or only ERIC documents dealing with reading.

Often, data bases that a particular researcher may wish to use have been prepared elsewhere with each having different input conventions and formats. For example, one data base might have been prepared with a logical record length of 100 and in EBCDIC code, utilizing both upper and lower case characters, while another might have been prepared with a logical record length of 72, in ASCII code, and be in upper case only with capitals indicated by a "/" preceding the capitalized letter. Version I of the L.A.P. will be able to handle input formats and sets of conventions that the researcher can specify (See 3.5.0 Translate Module).

A researcher may want to use output from one step in the modular L.A.P. execution as input to another. He may want to select subsets of a given data base, process each separately, then cross reference the results or subsets of the results. He may want to do transformations on the data

as it is being processed, and use the transformed data as input. Version I of the L.A.P. will be able to save output for further processing and will be able to save subsets of data once they are selected, in order to save the expense of repeated retrieval processing. The researcher will be able to present the system with a new file, or retrieve and use a file either he or another researcher has previously used or created (see 3.4.0 Input-Output Module).

1.4.0 Modes of Operation

In addition to the flexibility of modularity, input formats, and file handling, Version I of the L.A.P. will take advantage of the best features of two basic kinds of operation.

An interactive, "conversational" processing environment provides the user high flexibility with little training. He interacts with the computer by answering questions, providing the program with information about options he intends to implement for a particular run. On the other hand, a non-interactive, "batch" processing environment is significantly less expensive, because it can take advantage of "slack time" on the computer, and doesn't require costly telephone connect time. For example, one Los Angeles service bureau[3] priced interactive time at $360 per CPU hour, while batch processing was $150 per CPU hour. Language analysis processing requires considerable CPU time, since most computers are not designed for text scanning and string manipulation, but rather for numeric processing. Version I of the L.A.P. will provide interaction to collect the parametric information required for the run from the user, and batch

[3] C & C Computing, 8939 S. Sepulveda, Los Angeles, California

processing for the remainder of the run on the data base. It will do so by having an interactive module which sets up the parameters for the batch run (see 3.1.0 Interact Module and 3.2.0 Control Module).

1.3.0 Definition of Terms

Before proceeding into a detailed description of the System Design for Version I, several major terms must be defined.

Structurally, the I.A.P. will be made up of Modules, each of which will consist of a Program, Sub-Program or Routine.

A Program will perform multi-task operations. It will be made up of a number of Sub-Programs which in turn may contain several Routines each. Programs will produce complete, finished output to the user. An example of a Program is the Retrieve Module.

A Sub-Program will perform single-task operations of a complex nature. It will be made up of several Routines, each processing a portion of the Sub-Program's task. Output from a Sub-Program may be complete and go to the user, or may serve as input to another Sub-Program within a Program. An example of a Sub-Program is the Morphological Analysis Module

A Routine will perform single-task operations of a simple nature. It will usually be a dependent part of a Sub-Program or a Program, although occasionally it may stand alone (as in the case of Interface Modules which are single Routines). Output from a Routine will be used as input data or parameters for other Routines or Sub-Programs. It will provide no output to the user directly. An example of a Routine is the Translate Module.

There will be two basic types of Modules used in Version I of the L.A.P., Function Modules and Interface Modules.

A Function Module will be either a Program, Sub-Program, or Routine which performs an often complex operation. Since each Function Module will be "un-pluggable", it is important to think of the Module in terms of the function itself rather than the way in which it is accomplished. Otherwise, it will be difficult to establish a sufficiently generalized Interface Module allowing any other Program, Sub-Program or Routine to be plugged in, as long as it performs the same function.

An Interface Module will be a Routine linking the Control Module with a particular Function Module. While the Function Module is designed to be readily "un-pluggable", the Interface Module is designed to be a more permanent part of the package skeleton. It will not be un-plugged and replaced each time its associated Function Module is replaced, but only if the conceptualization of the function itself changes.

1.6.0 Documentation Concepts

Documentation of the L.A.P. will be, like the package itself, modular. If a function module of the package is changed, because a better program has been obtained for that function, the documentation can be un-plugged along with the software and as easily replaced.

There will be a series of documents reflecting the growth of the package as a whole, and of each of its component parts. There will be three basic types of documentation associated with various stages of developmental progress. They are: Design Documentation, User Documentation, and Product Documentation.

Design Documenation will consist of the L.A.P. System Design Document for each version of the package, together with the Module Design Document for each module associated with that version. Design documentation will be produced before the actual implementation of the package or any specific module. The System Design Document will contain a systematic conceptual overview of the capabilities of a particular version of the L.A.P., together with generalized descriptions of the modules to be incorporated in that version. There will be a new System Design Document for each successive update of the package as a whole. The intended audience for the System Design Document will be the general researcher who can find ways of making use of the package. The Module Design Document, on the other hand, will be of a more technical nature and will be aimed primarily at the programming staff whose responsibility it is to implement the design in a workable form.

User Documentation will consist of three basic types of documents: Module Development Announcements, User's Manual Materials, and Training Materials. Such documentation will be produced after a particular module is in operating condition and available to researchers for their use. As each function module is brought into working order, a Module Development Announcement will be released, indicating the functional capabilities of

the module, how it might be used in educational research, what kind of input it requires, and what kind of output it produces. At the same time, a User's Manual and Training Materials will be issued which will give a researcher the detailed information he needs to actually prepare data and submit a request for processing by the module.

Product Documentation will consist of detailed technical documentation, including flowcharts and actual program code for each of the modules. Such documentation will be released after all other documentation relating to the module, and will be intended for a technical audience only.

All three basic types of documentation will follow the same general format and be tagged with an appropriate numeric identifier which relates back to the original System Design Document for the version of which it is a part (see Documentation Outline, p. i). Below is an outline which indicates the general form of one such piece of documentation. It is an outline of a Module Design Document associated with Version I.

TITLE: DESIGN DOCUMENT: SCAN MODULE - L.A.P. VERSION I

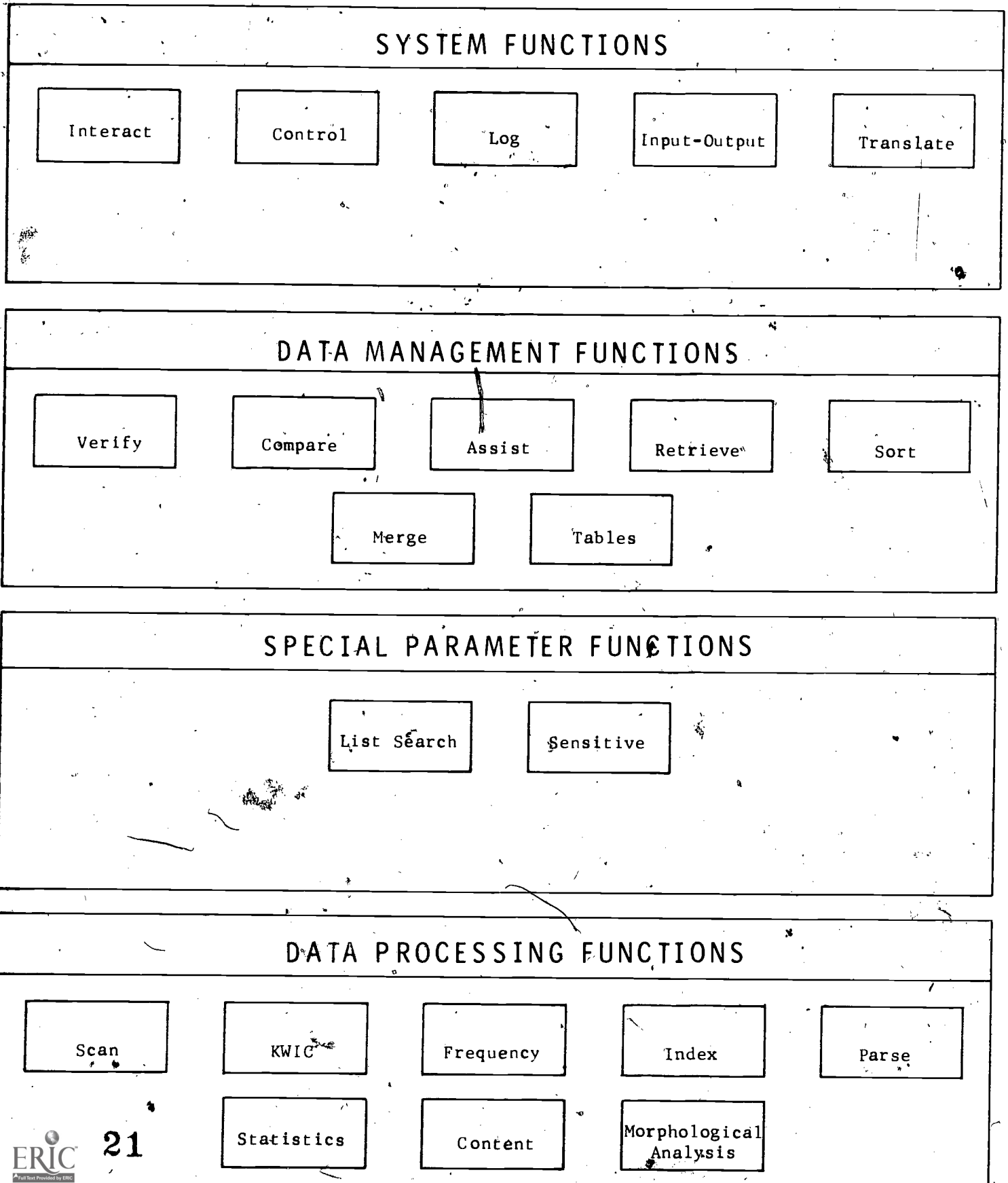
ABSTRACT

This is one of a series of technical design specifications for individual modules in the Language Analysis Package (L.A.P.).

1. Program Objective
2. Constraints and Limitations
3. Options and Defaults
4. Data File Specifications
   - Input
   - Output
5. Significant Algorithms
6. Significant Variables (Arrays, etc.)
7. Error and Other Messages
8. Called by and/or Calls

FIGURE 1

2.0.0 OVERVIEW OF L.A.P. DESIGN



2.1.0 Package Software Functions (see Figure 1)

The Language Analysis Package will provide four basic functions. They will be: System Functions, Data Management Functions, Special Parameter Functions, and Data Processing Functions.

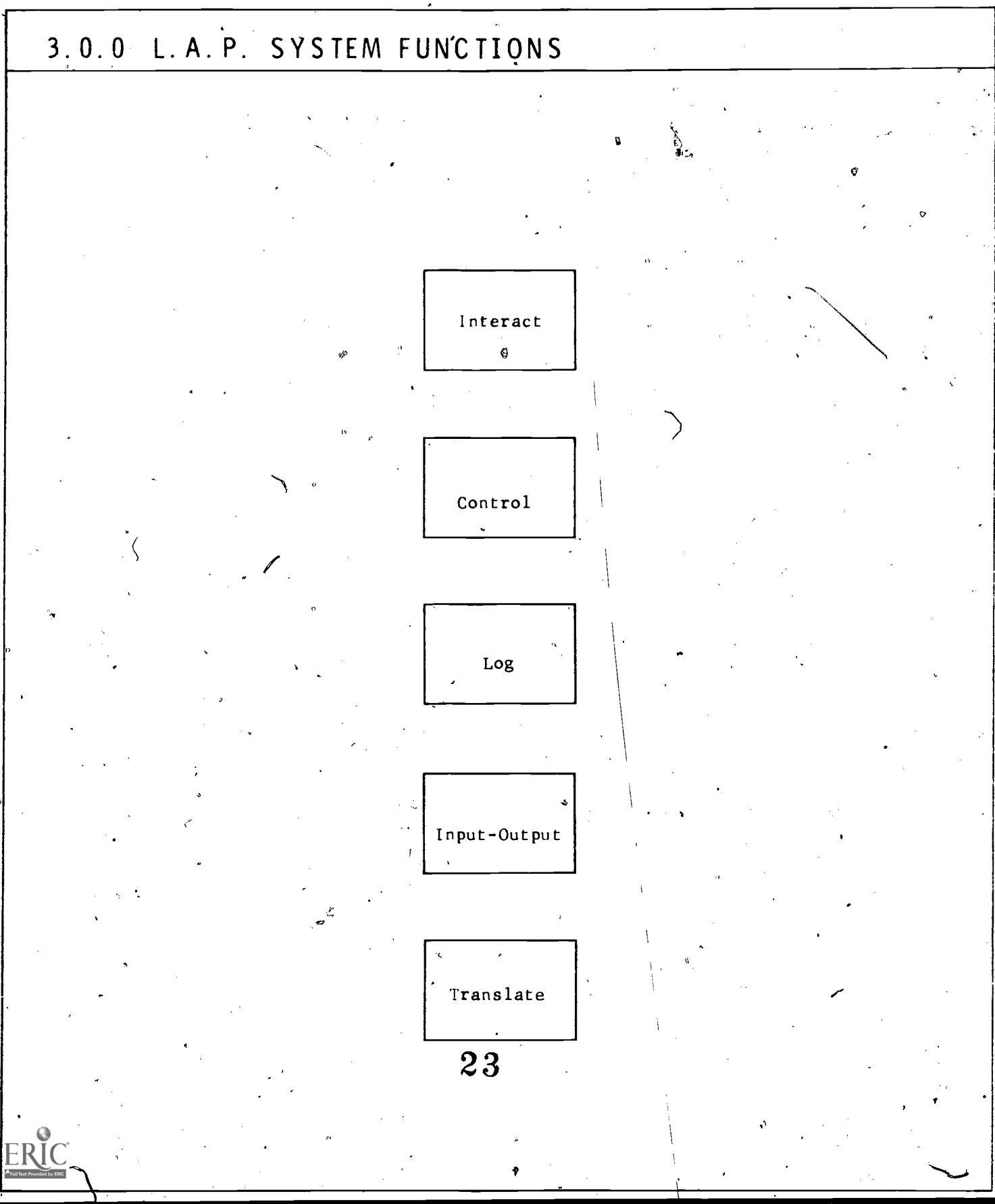
The System Functions will be concerned with the internal functioning of the package itself and the interface of the package with the user (see 3.0.0).

The Data Management Functions will be concerned with assisting the user in the preparation of his input data, data base control, and arranging his output in a form that will best facilitate his research (see 4.0.0).

The Special Parameter Functions will be primarily concerned with the definition of limitations on portions of the data to be processed, or unusual applications of package processing modules (see 5.0.0).

The Data Processing Functions will be concerned with the actual analysis of the input data, and the production of information that will further the user's research effort (see 6.0.0).

FIGURE 2


3.0.0 L.A.P. SYSTEM FUNCTIONS
Interact
Control
Log
Input-Output
Translate

3.0.0 OVERVIEW OF SYSTEM FUNCTIONS (see Figure 2)

System Functions will be those functions performed by the package which are primarily concerned with the internal functioning of the package itself and the interface of the package with the user. Unlike the Data Processing Functions, the System Functions will be written specially for the L.A.P., rather than obtained from programmers in the field of language processing. They will be tailored to the user community served by the package and be developed in a manner allowing easy modification to conform to the specific needs of new user requirements.

In addition, a high level of flexibility will be built into each of the System Functions so that a systems analyst may constantly and easily update the manner in which communication with the user takes place. In this way, the System Functions will be highly responsive to the modes of interaction most comfortable to the user community the package is serving at any given time.

For Version I, five modules will perform System Functions. They are: Interact Module, Control Module, Log Module, Input-Output Module, and Translate Module.

The Interact Module will help the user to set the parameters for his particular use of the package, and to set up the required Transaction Language Control necessary to obtain the output he needs.

The Control Module will function internally to take the Transaction Language Control and establish a decision table for use by the package in setting up a priority queue for accessing modules and routines for the present run.

The Log Module will provide the user with a record of his run, including such information as the control parameters used, and the action taken by the package based upon these parameters, as well as the date, costs, etc. of the run. In addition it will give the systems analyst information to assist in further optimizing the package as a whole.

The Input-Output Module will handle specifications made by the user concerning input and output files and provide him with a record of I-O operations.

The Translate Module will utilize information provided by the user concerning the form of his input data, and change the input constraints required of the processing module to a form compatible with the user's data. It will provide him with a record of such translations, and/or error messages, if his input specifications are incompatible with the processing he has requested.

3.1.0 Interact Module

The function of the Interact Module will be to act as an interface between the user and the package. Interact will run as a front-end portion of the total package, and prepare user control parameters to be appended to the input data which will then be run in batch mode at the main facility.

It will ask the user questions about the parameters of the run he is initiating and will utilize his answers to prepare computer compatible control parameters to be read by the Control Module.

Since it is a separately functioning entity, it will serve as a training program for initiating researchers into the use of the package. As each control parameter is compiled through the question and answer process, a correctly formatted Transaction Language Control statement will be printed out. The control statements also will be passed directly to the Control Module. The control statements will be output in a form appropriate for use as input to the main package programs, such as magnetic tape.

After the researcher has used the Interact Module for a while he may find that he is sufficiently familiar with the requirements of the Transaction Language Control to prepare his own control statements without computer assistance. Certainly, such user expertise is one of the goals of the Interact Module. As a teaching, as well as a functional program, it will keep a running count of user success and failure in the question answering process, and output such information to a systems analyst when polled. The systems analyst will use such information to modify and upgrade the package for maximum success within the environment of the actual researchers making use of the system.

3.2.0 Control Module

The function of the Control Module will be to read a set of control statements containing run parameters, and to perform a decision making function for that run.

In any particular case, the user will specify which package functions he wishes to use, as well as his particular output specifications. For example, he might wish to produce a KWIC, a rank-ordered frequency count and a parsing of his text. He will indicate his needs by means of Transaction Language Control statements which preceed his input data. These TLC statements will be produced either by use of the Interact Module, or (in the case of a sophisticated user) directly. The Control Module will read the statements and compile a decision table which can be used by the program during execution to determine which Function Modules will be called in which order for that run.

If the "sophisticated" user has made syntax errors in his preparation of the statements, the Control Module will print error messages which will help him correct his errors before re-submitting the job. The Transaction Language will be designed so that typical errors, such as the omission of a comma, will be automatically corrected, allowing execution to proceed. For such corrections, a message will be printed on the output indicating the assumptions made by the Control Module, helping the user to verify that the Control Module has not misunderstood his intent. The user will have the option to specify that he does not wish execution to proceed if assumptions were necessary.

In Version I, the user will specify the form of his input data, such as record length, order and location of variables (for Statistics Module) and estimated size of data base being used. In later Versions, the Control Module will scan this information and set parameters for

the run to optimize usage of computer equipment and peripherals. Such parameters will control selection of I-O procedures, storage and access procedures, etc. Messages will accompany the output, indicating the options used in a particular run. Provision will be made for user override of the defaults.

The user will also indicate special conventions used in his data, such as a slash preceding a letter to indicate upper case, or an "@pp" preceding a character stream to indicate the beginning of a new paragraph. The Control Module will evaluate the form of the input data, and decide if sufficient information is present to allow the requested function modules to execute. If execution is possible, it will determine whether the Translate Module needs to be called. Messages will be output to the user indicating missing information which prevents execution. As in other cases, translation parameters for the particular run will accompany output. If the Translate Module is needed, the Control Module will provide the input convention information contained in the TEC statements. During execution, the Control Module will use the decision table to access appropriate Interface Modules in an appropriate order.

Any or all of the Interface Modules can be called upon by the Control Module, and the order and structure of the calling procedure need bear no resemblance to the linear thinking of the user but can be structured in terms of machine efficiency for combination and ordering of functions required by the particular run.

3.3.0 Log Module

The function of the Log Module will be to provide the user with a summary of the processing in the present computer run. It will provide the systems analyst with information concerning those subsections of the package getting the heaviest usage, allowing appropriate optimizations.

The following information will be included on a Version I user log sheet:

- The user's identification (name, cost center, etc.)
- The date and time
- A listing of the TLC statements used
- A listing of modules and routines called upon
- Execution time
- Input and output devices utilized

The following information will be included on a Version I systems analyst log sheet:

- All the items found on a user log sheet.
- Breakdown of in and out times for each procedure
- Size of data base
- Later versions will also show such things as internal storage allocations used

3.4.0 Input-Output Module

The function of the Input-Output Module will be to allow the user to specify which devices will handle his input and output, and to provide the Log Module with a record of the I-O operations. It will

work in conjunction with the Control Module, as a Sub-Program. Default input will be from punch cards, and default output will be to high speed line printer. Input may also come from magnetic tape, or disk. The Input Module will also be used when input for one module consists of output from another (such as the Retrieve Module).

All output will normally go to the printer; however, users may have particular needs or preferences and may elect to use another output device. For example, the user may wish to save his output in some computer compatible form for later input to some other program. If so, he will specify output to magnetic tape, or punched cards.

Another, although complex use of the output module, will occur when the user wants his output to serve as input for another module within the package itself. Here, the output module not only will put the output onto a selected device, but also will put the data into an appropriate storage location within the system.

3.5.0 Translate Module

The function of the Translate Module will be to provide the interface between the user's data and the data conventions required by the particular modules he wishes to use. Like the I/O Module, it may be viewed as a subprogram of the Control Module.

Since it is extremely costly to convert a large data base to another format, the Translate Module will work in the opposite direction, converting the relatively few program conventions to the format in which the data exists. Such a conversion will be accomplished in the following manner:

Each of the modules will have an array associated with it which is accessible to the Translate Module. The arrays will each have a dimension of 256, corresponding to the 256 possible 8-bit codes. Each position in the array will hold an octal number equivalent to the new value (the data dependent value) which should be utilized by the particular Function Module for the current run. The Translate Module will set up the arrays for those Function Modules being called by the current run, using the old value (the Function Module dependent value) as a subscript to locate the appropriate position within the array into which to store the data dependent value. For example, if the KWIC Module is written to expect a slash ("/") preceding each character which is to be taken as upper case, and the user's data has been prepared with a dollar sign serving the same function, an octal 133 (equivalent to an EBCDIC "$") will be placed in position 97 of the array associated with the KWIC program, since 97 is the EBCDIC decimal equivalent of a slash ("/").[3]

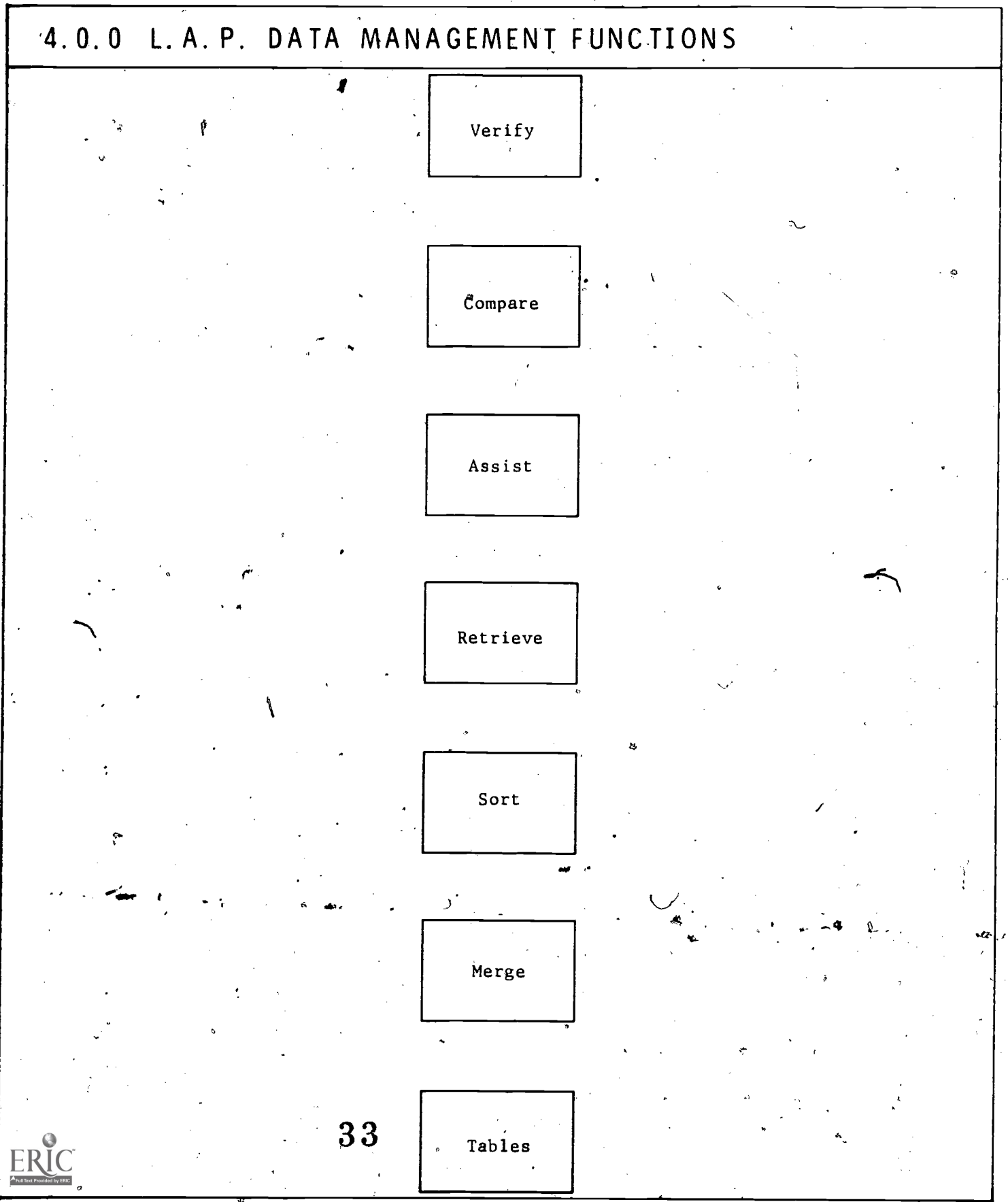
Each of the modules will have a specially prepared subroutine that initializes each of the program dependent variables (such as a variable "capital") to the value contained in the appropriate position in the array associated with that module.

---

[3] EBCDIC codes have been used, since the UCLA computer facility is a likely one on which to set up the package. The same algorithm could be used with a computer which uses ASCII, with an octal 040 being placed in position 47 of the array.

The utilization of the general purpose array will allow great flexibility when a new Function Module is to be added or substituted in the system, since a completely different set of input requirements may be accomodated without modification of other modules in the package.

FIGURE 3


4.0.0 L.A.P. DATA MANAGEMENT FUNCTIONS
Verify
Compare
Assist
Retrieve
Sort
Merge
Tables

4.0.0 DATA MANAGEMENT FUNCTIONS (see Figure 3)

Data Management Functions are those functions performed by the package which will be primarily concerned with assisting the user in the preparation of his input data, and arranging his output in a form which will best facilitate his research. Seven modules will perform Data Management Functions. They will be Verify Module, Compare Module, Assist Module, Retrieve Module, Sort Module, Merge Module, and Tables Module.

The Verify Module will provide the user with tests of keypunching accuracy, and will flag obvious errors such as the use of "L" instead of "1" within a numeric string.

The Compare Module will test one text against another and will flag differences which occur.

The Assist Module will allow the user to have the computer identify and flag, in his input, specific items on which he needs to take special action by hand coding.

The Retrieve Module will give the user the capability of pulling out a smaller section of a large data base according to specified criteria which apply to that section and not to other sections of the total data base. Such a retrieved subsection can be used as input to other modules of the package.

The Sort Module will rearrange the output from a processing module into an order defined by the user.

The Merge Module will rearrange the output from various previous processings into a single output in a form defined by the user.

The Tables Module will prepare output in tabular form, according to row and column specifications given by the user.

4.1.0 Verify Module

The function of the Verify Module will be to check on keypunching accuracy where errors might invalidate research results. The user will be able to specify in a TLC option statement one or more of the following data verifications:

- Numerics imbedded in alphabetic strings.
- Alphabetics imbedded in numeric strings.
- Unmatched parentheses or other paired delimiters (user must specify what delimiters).
- Strings greater than 16 characters in length.
- Use of "illegal" characters (user specified).

Default will be for the program to check all the items. Output from the module will be in the form of location information and type of error, with enough context given to clearly identify the error for correction.

4.2.0 Compare Module

The function of the Compare Module will be to allow the user to obtain information on the correspondences between two texts. Output will be in the form of an alphabetical list of words appearing in text one with their locations, followed by a list of words in text two which appear at the same relative locations, but are different from those appearing in text one.

Such lists will be headed with the user supplied identifier for each of the texts compared, and the date the comparison was made. An overall consistency quotient will be given, which is the ratio of identical words/total words.

4.3.0 Assist Module

The function of the Assist Module will be to allow the user to have the computer identify and flag in his input specific items on which he wishes to take special action by inserting hand coded information.

The user will specify alphabetic or numeric strings which he wishes to have identified and flagged, and he will identify one of two ways in which he wishes them flagged for his special handling (either by the insertion of 10 blanks immediately following the item, or by the insertion of a user defined special flag symbol).

The Assist Module will scan the text for the specified strings and produce a new text file with flags included. By use of the Input-Output Module, the user can specify the device on which the file is to be saved. Default is to print the file on the line printer. The user may specify the number of copies of output he wishes.

4.4.0 Retrieve Module

The function of the Retrieve Module will be to allow the user to search a sequential or an inverted file using either a Boolean combination of terms, or a list of numeric identifiers and to obtain subsets of information contained within that data base.

A sequential file is one arranged in "normal" order, with a series of sets of related information following each other sequentially. An example of such a file would be a personnel data base that would contain a sequence of sets of information such as name, address, education, salary, date employed, etc. for each employee.

An inverted file is similar to an index in many ways. It is arranged by sub-category, and lists the location of all occurrences of information relating to the sub-category. For example, such a file would have an entry for salary = $15,000 and would list the locations of full references on all employees with that yearly income.

A search utilizing a Boolean combination of terms could yield such information as all those references where AGE = 30 and EDUCATION = MA OR PHD AND SALARY = $12,000.

4.5.0 Sort Module

The function of the Sort Module will be to allow the user to specify particular regular orderings for his input or output.

Version I options will include:

- Ascending (i.e. A-Z or 1-1,000).
- Descending (i.e. Z-A or 1,000-1).
- Combination (i.e. several sort fields, with different options for each).

Later versions will include an option for reverse word (ex. all words ending in d together, with ed words before id words, etc.)

The user will specify the fields on which he wishes sorts to be performed, with the default being one field only consisting of the first 16 characters of the record. He will specify the type of sort he wishes. Certain modules which regularly require sorted output will have their own defaults. For example, KWIC output will be sorted in ascending alphabetical order on keywords, and ascending numeric order on locations if keywords are the same; frequencies will be sorted in ascending alphabetical order on the word field if the "Alpha" option is invoked, and descending numeric order on the frequency field if the "Rank" option is invoked.

4.6.0 Merge Module

The function of the Merge Module will be to rearrange the output from various previous processing steps into a single output. The most common use of this module will be for the production of merged KWIC and merged Tables, although it may also be used with such modules as the List Search Module to help the user establish larger dictionaries.

4.7.0 Tables Module

The function of the Tables Module will be to allow the user to reformat information gathered from the Data Processing Modules, in order to make relationships within the output more easily apparent.

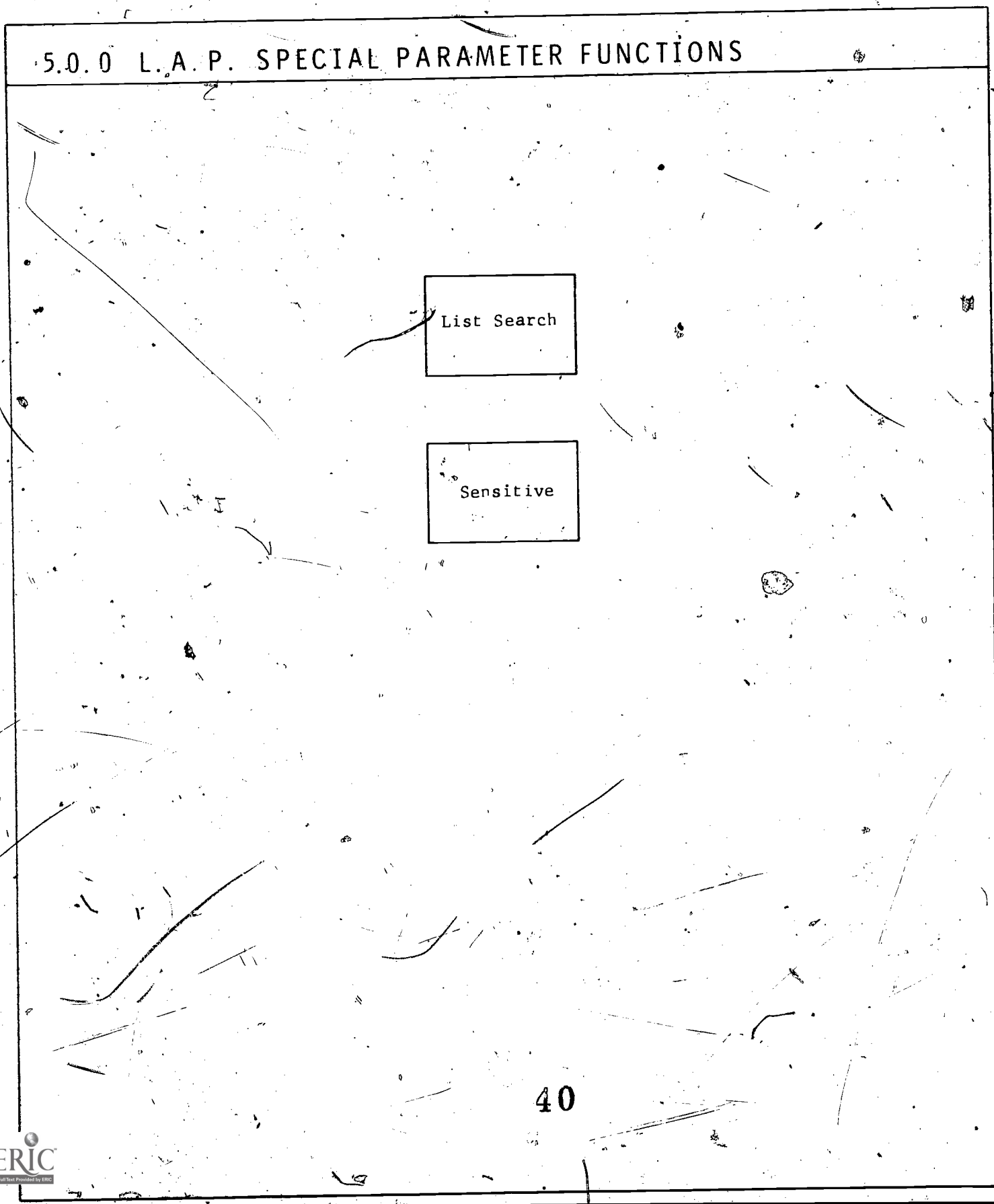
For example, if the user has obtained frequencies and percentages of numeric codes by using the Frequency Module, he will be able to make a table where closely related codes appear in vertical rows and the sub-categories of those codes appear in horizontal columns.

In the example below, codes 10-19 appear in row 1, 20-29 in row 2, and 30-39 in row 3. Relationships between the number of occurrences and percentages of codes 15 and 25 may become clearer by looking at column 5 (subcode 5).

	SUBCODE 0		SUBCODE 1		SUBCODE 2		SUBCODE 3		SUBCODE 4		SUBCODE 5		SUBCODE 6		SUBCODE 7		SUBCODE 8		SUBCODE 9	
	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%	OCC	%
CODE 10																				
CODE 20																				
CODE 30																				

Version I of Tables Module will allow the user to indicate appropriate headings for his tables, and to specify which values shall appear in row and column positions.

FIGURE 4


5.0.0 L.A.P. SPECIAL PARAMETER FUNCTIONS
List Search
Sensitive

5.0.0 SPECIAL PARAMETER FUNCTIONS (see Figure 4)

Special Parameter Functions will be those functions performed by the package which are primarily concerned with definition of limitations on portions of the data to be processed, or unusual applications of package processing modules. In version I two modules will perform Special Parameter functions. They are: List Search Module and Sensitive Module.

The List Search Module will allow the user to define lists of words or phrases which shall be either included or excluded from the processing requested.

The Sensitive Module allows the user to specify particular contexts within which, and only within which, processing is to be done.

5.1.0 List Search Module

The function of the List Search Module will be to allow the user to specify at run time particular strings (alphabetic or numeric) which shall be either included or excluded from the processing. Such sets of strings will be input in the form of a list which preferably will be alphabetized, and which will be preceded by a system control card specifying INCL or EXCL.

INCL. Specifies that processing shall be done only on strings which appear in the following list. All other strings shall be ignored.

EXCL. Specifies that processing shall be done on all strings except those which appear in the following list. Those strings in the list shall be ignored during processing.

The INCL - EXCL option will allow the user to optimize production costs by processing only the data which is essential to his particular application. The INCL - EXCL option should be used with care, especially in conjunction with Sensitive, Content and Parse since errors can occur if data pertinent to these processors is omitted through excessive limitation of the data to be included as input.

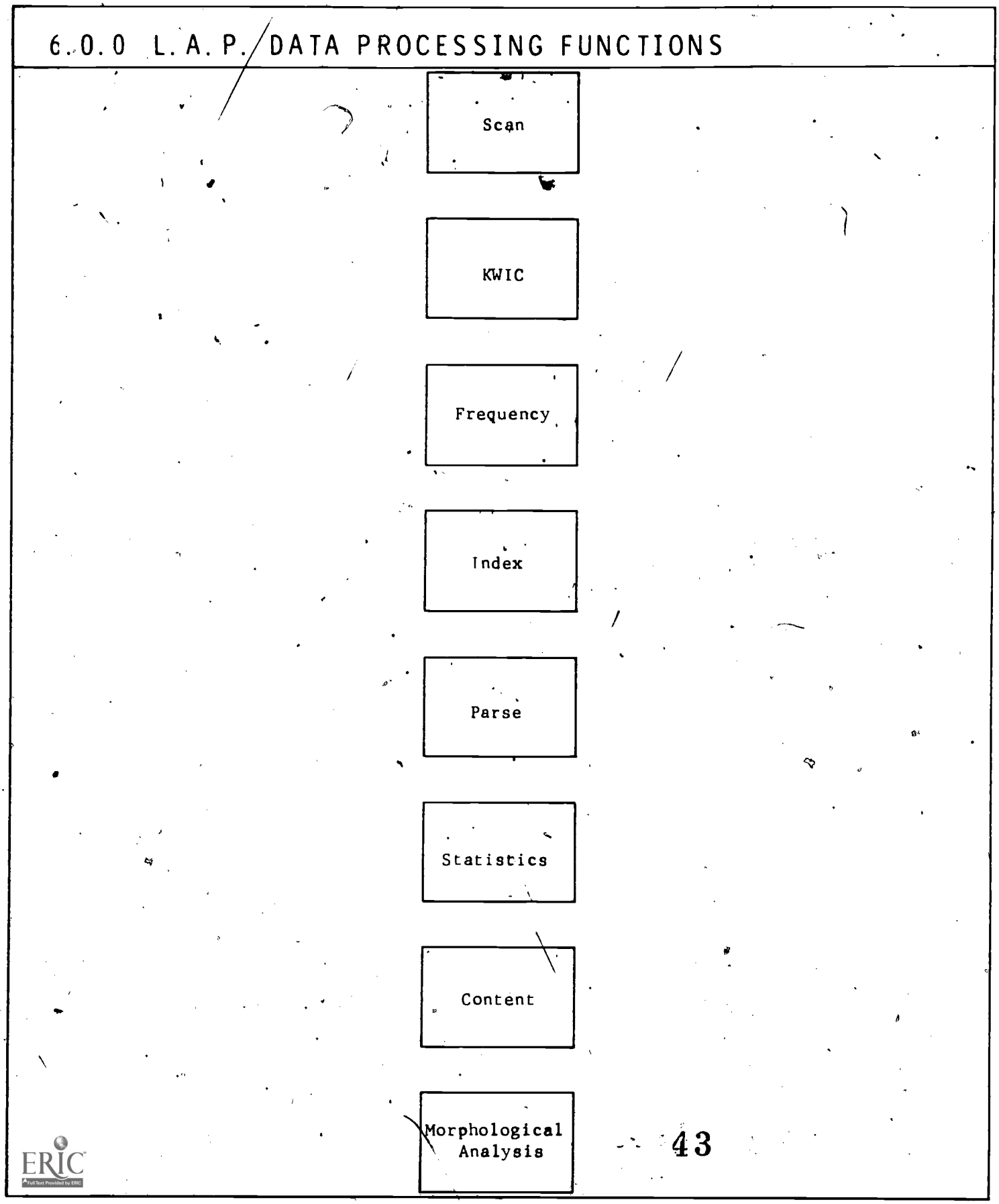
5.2.0 Sensitive Module

The function of the Sensitive Module will be to allow the user to define adjacent contexts which specify parts of the input text to be processed. For example, a user might want to process the word "reading" if and only if the word "remedial" preceded it. Or he might wish to process only the words occurring within the Title of a library citation.

The user will specify:

- The string (alphabetic or numeric) which provides the Key for processing.
- The string (alphabetic or numeric) which provides the sensitive context.
- The maximum distance (in number of characters or number of words) from the Key within which the sensitive context must occur.
- The direction (left or right) from the Key in which the sensitive context must lie.

FIGURE 5


6.0.0 L.A.P. DATA PROCESSING FUNCTIONS
Scan
KWIC
Frequency
Index
Parse
Statistics
Content
Morphological Analysis

6.0.0 DATA PROCESSING FUNCTIONS (see Figure 5)

Data Processing Functions will be those functions performed by the package which will be primarily concerned with the actual analysis of the input data, and the production of information which will further the user's research effort. Eight basic modules will perform Data Processing functions. They are: Scan Module, KWIC Module, Frequency Module, Index Module, Parse Module, Statistics Module, Content Module and Morphological Analysis Module.

The Scan Module will be the heart of the data processing functions. It will read text in, scan it character by character, and establish word boundaries, etc.

The KWIC Module will produce a Keyword in Context (Concordance) listing of the data, allowing the user to specify any one of two basic output formats, and the desired length for the before and after contexts.

The Frequency Module will produce information on the number of occurrences and percentages for words, phrases, and codes within the text. The user may specify alphabetical ordering or rank ordering.

The Index Module will produce an alphabetical index of the locations of references to particular words, phrases or codes. The user will be able to specify the terms in which locations will be given.

The Parse Module will provide the user with information concerning the grammatical structure of his input text, in the form of a tree arrangement of surface and deep structures.

The Statistics Module will produce numerical analyses of such items as means and standard deviation of word length, sentence length, paragraph length and numerous others.

The Content Module will allow the user to develop semantic categories, and test his data against them.

The Morphological Analysis Module will allow the user to obtain listings of root words contained within his text. It will be used primarily in conjunction with other processing modules, such as Parse.

6.1.0 Scan Module

Version I of the Scan Module will function as a "front end" for several of the processing modules (KWIC, FREQ, etc.). Since it will be heavily used, and since it performs its decision making function by scanning character by character through the input text, special attention will be paid to problems of optimization. Its major objective is threefold. It will read in data as necessary from the input stream; it will find and return to the calling program the beginning and end points of a word; and it will determine if the delimiter(s) following a word is a "special" character which signals the necessity of some kind of special handling involving an additional subroutine call.

Depending on the calling module, Scan may return either the beginning and ending points of the word, or the word itself for storage in an array.

The user will be able to specify a definition of characters to be considered as valid word-parts. Default will be to have the alphabet, apostrophe, hyphen and numbers considered as word-parts with all other characters considered as delimiters. In addition, the user will be able to specify which of the delimiters should be considered as "special" characters.

6.2.0 KWIC Module

The function of the KWIC Module is to produce Keyword-in-Context (Concordance) listings from the input text. Location information will be broken down into four main categories: level 1 (ex. Document ID), level 2 (ex. Page), level 3 (ex. paragraph), and level 4 (ex. line). The KWIC Module will be run frequently in conjunction with the List Search Module. One such application will be when producing KWIC's on library citations where only words in the title will be processed.

The user may specify the following:

- Length of before context to be specified either in number of words or number of characters. (default is 48 characters)
- Length of keyword. (default is 16)
- Length of after context to be specified either in number of words or in number of characters. (default is 48 characters)
- Sort fields and ordering (default is keyword, location, in ascending order)
- Either of the following two output formats:
    A) one line, keyword centered
    B) two line, keyword right justified

  (default is A)

6.3.0 Frequency Module

The function of the Frequency Module will be to produce alphabetical and rank ordered computations of the number of times words and/or numeric codes occur, and what percentage of the total text is represented by each.

The user may specify the following:

- Alphabetical ordering of the list. (words beginning with "A" first)
- Rank ordering of the list. (most frequent first)
- Occurrences only.
- Percentages only.

Default will be for the Module to produce all four. The Frequency Module will be run often in conjunction with the Tables Module to produce a tabular presentation of the information in a user specified format.

6.4.0 Index Module

The function of the Index Module will be to allow the user to compile an index of locations of specific words and phrases within his text. The module will generally be used in conjunction with the List Search Module.

The index produced will be alphabetically sorted. The default for locations will be the same as for the KWIC Module (see 6.2.0). If no inclusion or exclusion list is provided by the user, a standard exclusion list will be used which consists of high frequency words such as "and", "the", "in", "by" etc.

6.5.0 Parse Module

The function of the Parse Module will be to provide the user with information concerning the grammatical structure of sentences contained within the input text.

The user may provide his own grammar and dictionary in a format compatible with the program parameters.

Output is in the form of tree-structures representing the surface and deep structures of the sentences.

The Parse Module will be run in conjunction with the Morphological Analysis and List Search Modules.

The Parse Module should be used with care, since processing costs for large input texts are often high.

6.6.0 Statistics Module

The function of the Statistics Module will be to allow the user to obtain summary information of a statistical nature concerning his input text.

Following are some of the more obvious statistical measures of text:

- Total number of occurrences of each mark of punctuation.
- Total number of words in text.
- Total number of different words.
- Ratio of unique/total words (type/token ratio)
- Frequency of words of length n (where n ranges from 1 to 24 characters).
- Frequency of sentences of length m (where m ranges from 1 to 40 words).
- Total number of sentences in text.
- Total number of paragraphs in text.
- Mean word length in sentences.
- Mean sentence length in words.

- Mean paragraph length in sentences.
- Standard deviation of word length.
- Standard deviation of sentence length.
- Standard deviation of paragraph length.
- Third moment of word length.
- Fourth moment of word length.

6.7.0 Content Module

The function of the Content Module will be to allow the user to set up dictionaries of phrases in numerous semantic categories, and to match his particular text against one or more of these dictionaries and have a category item analysis performed. A tabulation will be given by category of the matching items and the number of occurrences. For example, a user studying the relationships of Mexican American oriented state approved text books may want to make phrase dictionaries for semantic categories such as "nuclear family", "extended family", "individualism", "socialization", "present orientation", "future orientation", "patriarchy", "matriarchy", "cooperation", and "competition". By using the Content Module, he could determine the concept weighting which occurs.

6.8.0 Morphological Analysis Module

The function of the Morphological Analysis Module will be to allow the user to obtain information concerning the root words contained within his input text. Generally, this module will be run in conjunction with other modules such as Parse and List Search but the user may obtain a root word list independently of other module uses.

7.0.0 TRANSACTION LANGUAGE CONTROL (T.L.C.)

The function of the Transaction Language will be to provide the user with a method of communicating parameters for operating the computer run. It will allow the user to specify the form of his input data, the package functions he wishes to use, specific options he requires, and the form of output he wishes to obtain.

7.1.0 General Concepts

The basic design of the TLC emphasizes ease of use. There are few syntax constraints. With the exception of the initial identifying "!" in column 1, parameters may be in a relatively free format form. After the "Control Word" (RUN, FUNC, OPT, etc.), required sub-items of information may be in any order. Imbedded blanks are ignored so the user may format his TLC cards as he wishes. If a command is misspelled, the system will make a "best guess" based on the context and the first two letters, and will print its "guess" or a message indicating that it has insufficient information to continue processing.

7.2.0 Required System Commands

All TLC cards will begin with a "!" in column 1. Continuation cards will be indicated by "!" in columns 1 and 2.

The following six "Control Words" will be required for any run made with the package. The Control Word must be the first item following the exclamation point, but need not be in any particular column.

- !RUN
- !INPUT
- !FUNC
- !OPT
- !OUTPUT
- !DONE

!RUN

The !RUN card will tell the system that a run is being initiated. Any information punched on the !RUN card from columns 6-80 or on continuation cards will be used as run identification and will be printed at the top of each page of output. No more than two continuation cards will be allowed.

!INPUT

The !INPUT card will specify any special parameters of the user's input. The following information must be included on the !INPUT card and/or its continuations:

- Input record length
- Input device (magnetic tape, cards, etc.)
- Code used (ASCII, BCD, EBCDIC)
- Special conventions

  ('PG = NEW PAGE; '/' = UPPER CASE; etc.)

Any number of continuation cards may be used.

!FUNC

The !FUNC card will indicate which Data Management and Data Processing functions the user wishes to employ. Following is a list of acceptable functions which may be specified by the !FUNC card:

- 'ERIFY
- COMPARE
- ASSIST
- RETRIEVE
- SORT
- MERGE
- KWIC
- FREQUENCY
- INDEX
- PARSE
- STATISTICS
- CONTENT
- MORPH AN[redacted]
- LIST SEARCH

One !FUNC card will be required for each function employed, and it must be followed immediately by its associated !OPT card. However, the two-card "sets" (!FUNC and !OPT) may occur in any order.

!OPT

The !OPT card immediately follows the !FUNC card and is associated with it. An !OPT card without a preceding !FUNC card will result in an error message and vice-versa. The !OPT card will associate a list of appropriate options with the functions requested. For example, one option of the KWIC function is the use of an inclusion or exclusion list; one option of the Frequency function is a rank-ordered list.

!OUTPUT

The !OUTPUT card will specify what output the user wishes to receive, and will allow him to direct his output to whatever device he desires. He may suppress Log output and summary diagnostics if he wishes. It is this card which will specify the destination if the output from one function (for example Frequency) is to be used as the input for another function (for example Tables).

!DONE

The !DONE card indicates that all TLC cards have been input and the user is requesting compilation and execution. All cards after the !DONE card will be considered input data.

8.0.0 SUMMARY AND CONCLUSIONS

The most important features of the Language Analysis Package will be adaptability of modular design and flexibility of user-defined options.

Individual portions of the package will be used without the user being penalized by high processing costs due to superfluous modules being called.

Development costs will be held to minimum by utilizing existing programs already written and debugged by specialists in the field of natural language processing.

Documentation also will be modular, and will develop organically with the growth of the package. At every stage documentation will be produced reflecting latest developments and most recent revisions.

The value of the L.A.P. will lie in the research opportunities it will open for the non-computer oriented researcher, who previously had no ready tool to do analysis of textual data.

Working Paper 2

DESIGN DOCUMENT: CONTENT MODULE - L.A.P. VERSION I (TN 5-72-35)

Ann Porch and Pat Lang

This document is one of a series of programming design specifications for individual modules of the Language Analysis Package (L.A.P.).[1] The section of the system design to which it is related is 6.7.0.

Program Objectives

Version I of the Content Module will function as a semantic content analysis module by allowing the user to construct any number of sorted dictionary files using phrases and/or single words.[2] Each dictionary file will represent a set of user-defined semantic categories. The program will "score" the input text by matching it against the dictionary and give the user the total number of different categories a particular word or phrase falls into, as well as the categories in which it falls.

---

[1] See TM 5-72-06, "Language Analysis Package (L.A.P.) Version I System Design," for an overview of the package.

[2] A modification of an existing program will be used for the Version I Content Module. The program is SCORTXT, developed by Gerald Fisher of the University of Connecticut. Full documentation on the program may be found in Fisher's "The SCORTXT Program for the Analysis of Natural Language," University of Connecticut, Bureau of Educational Research.

Constraints and Limitations

- When the text is scanned for dictionary matches, only the longest string will be matched. For example, if the dictionary contains both "not very much" and "not very," the text phrase "not very much" will be counted as matching only "not very much".

- No punctuation (except apostrophes and hyphens) may be included in dictionary phrases. Texts longer than 1,500 words must be broken into sub-texts if high-frequency words are category entries.

Options and Defaults

Options and defaults for the Content Module will be as follows:

- Print text in original form (Default = no print)
- Print text in array form (Default = no print)
- Print sorted dictionaries (Default = no print)
- Print a reduced text (Default = no print)
- Print an item analysis of each category (Default = print)
- User specified word length for text in array form (Default = 16)
- User specified input record margins for text (Default = 1,72)

Data File Specifications

Input files for Version I must follow the ordering shown in Appendix A. Dictionary input will consist of one or more dictionaries.

Output for Version I will consist of punched or printed output as shown in Appendix B.

Significant Algorithms

Category Indicator Algorithm:

After a dictionary is read in and sorted internally into alphabetical sequence, a bit string will be created for each dictionary entry with on-off indicators for each category represented. Thus if the dictionary has 1,000 entries which fall variously under five categories, then for each of the 1,000 entries a bit string of length five will be created with 1's in each category position to which the entry belongs. The sorted dictionary and the dictionary bit strings are added to the file DICT.

Significant Variables

There are no variables of special significance associated with the Content Module.

Error and Other Messages

The following messages are printed out by the Content Module:

- "Dictionary Not Found" if there is no dictionary associated with the input for a particular run.

- "End of Job" if the run terminates normally when there are no further texts to be read.

Called by and/or Call

The Content Module is called only by the Control Module.

The Content Module will call the following internal subroutines:

- ARRAY

- DINIT

- INDXDLM
- ITEM
- LSTAT
- MAKEDIC
- SORT
- PHRASE
- INTABLE

SWRL

COMPUTER SYSTEMS
FILE/RECORD LAYOUT

RECORDING MODE:
FIXED ☐
LENGTH ____
VARIABLE ☐
MAX ____
MIN ____

BLOCKING FACTOR:
BLOCK CONTAINS
☐ RECORDS

SEMA [illegible]

App [illegible] Input 1 Text [illegible]

CARD LAYOUT ☐
TAPE
RECORD LAYOUT ☐

FORMAT LEGEND:
△ = Blank
A = Alpha
N = Numeric
= Alpha/Num.
P = Packed Dec.
O = Octal
H = Hexadecimal
B = Binary

TITLE: Content Module DATE: May 8, 1972

FILE ID: Input

PROGRAMMER:

RECORD I.D. Text Header (one per text)	Text ID	User Comments	
FORMAT			
POSITIONS	5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100		
☐ CONTINUED RECORD I.D. Text (as many as needed)	text punched (029) within user specified card margins		
FORMAT			
POSITIONS	5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100		
☐ CONTINUED RECORD I.D. End of Text (more texts may follow)	**		
FORMAT			
POSITIONS	5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100		
☐ CONTINUED RECORD I.D. End of All Texts (Dictionary Selection Input must follow)	$$		
FORMAT			
POSITIONS	5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100		

59

SWRI COMPUTER SYSTEMS
FILE/RECORD LAYOUT

TITLE: Content Module DATE:

FILE ID: Input

PROGRAMMER:

RECORDING MODE
FIXED ☐
LENGTH
VARIABLE ☐
MAX
MIN
BLOCKING FACTOR:
BLOCK CONTAINS
☐ RECORDS

REMARKS:
Dictionary Input Input

CARD LAYOUT ☐
TAPE
RECORD LAYOUT ☐

FORMAT LEGEND:
△ = Blank P = Packed Dec.
A = Alpha O = Octal
N = Numeric H = Hexadecimal
X = Alpha/Num. B = Binary

RECORD I.D.
Dictionary Header
(one per dictionary)

Dictionary Name

Maximum number of characters in any entry

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.
Dictionary Entry
(as many as are needed)

Dictionary Entry

category no.

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.
End of Dictionary
(more dictionaries may follow)

**

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.
End of All Dictionaries
(Text Input must follow)

$$

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

SWRL

OUTLINE OF FILE RECORD LAYOUT

RECORDING MODE

FIXED

LENGTH

VARIABLE

MAX

MIN

BLOCKING FACTOR:

BLOCK CONTAINS ____ RECORDS

Dictionary Selection Input — Input

TITLE: Content Module DATE: May 8, 1972

FILE ID: Input

PROGRAMMER:

CARD LAYOUT

TAPE RECORD LAYOUT

FORMAT LEGEND:

B = Blank — P = Packed Dec.
A = Alpha — O = Octal
N = Numeric — H = Hexadecimal
X = Alpha/Num. — B = Binary

RECORD I.D.
Dictionary Selection
(as many as needed)

Dictionary Name

Dictionary ID no.

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.
End of All Dictionary
Selection (last input
record)

$$

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

61

SWRL COMPUTER SYSTEMS FILE/RECORD LAYOUT

TITLE: Content Module DATE: May 8, 1972

FILE ID: OUTPUT

PROGRAMMER:

RECORDING MODE:
FIXED [ ]
LENGTH
VARIABLE [ ]
MAX
MIN
BLOCKING FACTOR:
BLOCK CONTAINS [ ] RECORDS

REMARKS:

Appendix B Output 1
Score Result Output

CARD LAYOUT [ ]
TAPE
RECORD LAYOUT [ ]

FORMAT LEGEND:
b = Blank P = Packed Dec.
A = Alpha O = Octal
N = Numeric H = Hexadecimal
X = Alpha/Num. B = Binary

RECORD I.D.
Score Result (as many as needed - one output record per 18 categories)

FORMAT
POSITIONS

Text ID | category 1 score | category 2 score | category 3 score | category 4 score | category 5 score | number of categories is user controlled (up to 18 per record) | category 18 score | Dictionary ID no.

5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.
Text in Array Form (optional)

FORMAT
POSITIONS

Text ID | Total Number of words | Word Length of longest word | Text words printed in array form (number of columns each = 80/word length of longest word.)

5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.
Dictionary (one per dictionary)

FORMAT
POSITIONS

Dictionary Name | No. of entries | Maximum of Entry length | No. of categories | Entry specification 0=single word 1=phrases

5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.
Dictionary (Cont.) (as many as there are entries in dictionary)

FORMAT
POSITIONS

Dictionary Entry + bit string representing categories to which it belongs (Lengths of entry and bit string vary.)

5 10 15 20 25 30 35 40 45 50 55 60 65 70 80 85 90 95 100

62

SWRI

COMPUTER SYSTEMS
FILE/RECORD LAYOUT

RECORDING MODE:
FIXED ☐
LENGTH ____
VARIABLE ☐
MAX ____
MIN ____
BLOCKING FACTOR:
BLOCK CONTAINS ____ RECORDS

REMARKS:
Output 2

CARD LAYOUT ☐
TAPE
RECORD LAYOUT ☐

FORMAT LEGEND:
△ = Blank, A = Alpha, X = Numeric, - = Alpha/Num., P = Packed Dec., O = Octal, H = Hexadecimal, B = Binary

TITLE: Content Module DATE: May 8, 1972

FILE ID: OUTPUT

PROGRAMMER:

RECORD I.D. Reduced Text (optional)	Dictionary Name	Text ID	Number of words in text	Number of categories	
FORMAT					
POSITIONS	5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100				
☐ CONTINUED RECORD I.D. Reduced Text (con't)	Reduced Text (bit string for each word in text indicating all those categories to which a word belongs. The string will be filled in with zeros for a word not found in any category. If there is a match on a multi-word phrase, then the first word's bit string will indicate all the categories to which the phrase belongs, and the bit strings for all other words in the phrase will be filled with ones)				
FORMAT					
POSITIONS	5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100				
☐ CONTINUED RECORD I.D. Item Analysis (optional)	"ITEM ANALYSIS FOR DICTIONARY 'NAME'"				
FORMAT					
POSITIONS	5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100				
☐ CONTINUED RECORD I.D. Item Analysis (con't)	"COUNTS FOR CATEGORY 'NUMBER'"				
FORMAT					
POSITIONS	5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100				

SWRI

COMPUTER SYSTEMS
FILE/RECORD LAYOUT

RECORDING MODE:
FIXED ☐
LENGTH
VARIABLE ☐
MAX
MIN
BLOCKING FACTOR:
BLOCK CONTAINS
RECORDS

REMARKS:
Output 3

CARD LAYOUT ☐
TAPE
RECORD LAYOUT ☐

FORMAT LEGEND:
△ = Blank
A = Alpha
N = Numeric
∧ = Alpha/Num.
P = Packed Dec.
O = Octal
H = Hexadecimal
B = Binary

TITLE: Content Module DATE: May 8, 1972

FILE ID: OUTPUT

PROGRAMMER:

RECORD I.D.
Item Analysis (con't)

"CATEGORY NAME" "CATEGORY COUNT"
or "NO COUNT FOR THIS CATEGORY"

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED
RECORD I.D.

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED
RECORD I.D.

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED
RECORD I.D.

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

Working Paper 3

DESIGN DOCUMENT: KWIC MODULE--L.A.P. VERSION I (TN 5-72-37)

Ann Porch

This document is one of a series of programming design specifications for modules of the Language Analysis Package (L.A.P.). It is related to section 6.2.0 of the Version I system design document.[1]

Program Objectives

The function of the KWIC Module will be to produce keyword-in-context (concordance) listings from the input text. Such listings will contain location information which will be broken down into four main categories: level 1 (ex. Document ID), level 2 (ex. page), level 3 (ex. paragraph), and level 4 (ex. line). The KWIC Module will usually be run in conjunction with the List Search Module and inclusion or exclusion dictionaries. It may also run in conjunction with the Sensitive Module with user defined limitations on the portions of the input data to be presented.

Constraints and Limitations

There will be no constraints or limitations for input data for the KWIC Module, other than those described in design specifications for the Scan Module.[2]

---

[1]See TM 5-72-06, "Language Analysis Package (L.A.P.) Version I System Design."

[2]See TN 5-72-27, "Design Document: Scan Module--L.A.P. Version I."

Options and Defaults

Options and defaults for the KWIC Module will be as follows:

- Length of before context. The length of context will be specified in number of characters. (Default = 48)

- Length of keyword (Default = 16)

- Length of after context. The length of context will be specified in number of characters. (Default = 48)

- Form for identifying location information within the input text. (Default=@TXT for level 1; @PG for level 2; @SEC for level 3; and a computer generated count of input records (lines) for level 4)

- Sort field and ordering (Default = keyword, location in ascending order)

- Output format. There will be two choices: A.) one line, keyword centered, and, B.) two line, keyword right justified on line 1. (Default = one line, keyword centered)

Data File Specifications

Input

Data will be input to the Scan Module and will be considered as a stream of characters. No notice will be taken of record boundaries. Data which has been prepared utilizing other conventions can be handled by use of the TRANSLATE Module. The SCAN Module will be used in conjunction with the KWIC Module. It will break the text into words and check for special characters needed by sub-routines handling location information, etc. Parameters passed from SCAN to

KWIC will be text array identification, and the beginning and end point of the word.

Output

Output data will fall into one of two user specified formats. An example of each is shown in Appendix A. Output may be obtained on the high speed printer, punched cards, or magnetic tape.

Significant Algorithms

There are three significant algorithms connected with the KWIC Module. They are:

- The Three Record, Circular Read Algorithm (described more fully in "Design Document: Scan Module - L.A.P. Version I")
- The Contexting Algorithm
- The Location Information Algorithm

Three Record, Circular Read Algorithm. To find the context, the specified length of the before context will be subtracted from the pointer value for the beginning of the word. The result will be stored in a variable indicating the beginning point for the context.

To find the remaining portion of the context, the after context length will be added to the pointer value for the end of the word. The result will be stored in a variable indicating the ending point for the context.

A test will be made to determine if special action needs to be taken because values of either of the variables indicating beginning point and ending point for the context fall outside the bounds of the array. If so, special action will be taken as described later

below. If both variable values fall within the bounds of the array, the context may be output directly from the array itself to a file for later sorting. A standard, implied-Do type print statement may be used.

If the value of the variable indicating the beginning point of the context is negative, special handling is required. The negative value will be added to the upper bound of the array and the result stored in the begin-context variable. When the context is output to the file for later sorting, the print statement will have two parts. First the array values from the value of begin-context variable to the end of the array will be output. Then the array values from position 1 of the array to the value of the end-context variable will be printed.

If the value of the variable indicating the ending point of the context is greater than the upper bound of the array, special handling is also required. The array's upper bound value will be subtracted from the value of the end-context variable and the result stored in the end context variable. Again, when the context is output to the file for later sorting, the print statement will have two parts. As before, first the array values from the value of the begin-context variable to the end of the array will be output. Then the array values from position 1 of the array to the value of the end-context variable will be printed.

The Location Information Algorithm. Variables will be set up for each of the following four levels of location information:

- Level 1 (ex. document ID)
- Level 2 (ex. page)
- Level 3 (ex. paragraph)
- Level 4 (ex. line)

The user will specify a flag character (which must not be the same as any of the characters he has specified as valid word builders) and up to four following characters to indicate the function he is indicating. For example, the user may specify that all location information will be flagged with an '@'. He may flag pages with "@PG" and Paragraphs "@PAR".

The SCAN Module will pass to the Location subroutine the information that it has found a special character. The Location subroutine will then check to see if the special character found matched those which the user has specified as meaningful for the KWIC Module. If so, special action will be taken. Variables will be set up for values for each level. Level 1 (ex. document ID) identifiers will consist of three alphanumeric characters supplied from the input stream. (Default - TXT) All other variables will be counters. Counts for Level 2 indicators will be dependent on Level 1, and will be restarted whenever a new level 1 indicator is found. Likewise, counts for level 3 and 4 indicators will be dependent on level 2, and be restarted whenever a new level 2 is encountered.

For output, the contents of the four level variables will be introduced into the output stream in the positions appropriate to the

output format option chosen by the user.

Significant Variables

There are four significant variables in the KWIC Module. They are:

-Three Record Array

-Pointers to word beginning and end

-Pointers to context beginning and end

-Four variables for the levels of location information

Error and other Messages

The following messages are printed out by the KWIC Module:

- "End of Job" if the run terminates normally when there are no further texts to be processed.

Called by and/or Calls

The KWIC Module is called only by the Control Module.

The KWIC Module may call the following other modules:

- List Search (5.1.0) optional
- Sensitive (5.2.0) optional
- Sort (4.5.0) always called

SWRL COMPUTER SYSTEMS
FILE RECORD LAYOUT

Appendix A
Output

TITLE: _____

FILE ID: _____

PROGRAMMER: _____

RECORDING MODE: F = Fixed, V = Variable, U = Undefined
LENGTH: MAX ____ MIN ____
BLOCKING FACTOR:
BLOCK CONTAINS ____ RECORDS

CARD LAYOUT
TAPE
RECORD LAYOUT

FORMAT (USE): B = Blank, A = Alpha, N = Numeric, X = Alpha/Num.
P = Packed Dec., O = Octal, H = Hexadecimal, B = Binary

RECORD I.D.
Output option A

location before context keyword after context

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.
Output option B

location before context Keyword

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.
Output Option B cont.

after context

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

☐ CONTINUED

RECORD I.D.

FORMAT
POSITIONS 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100

Working Paper 4

OVERVIEW OF IDCMS DESIGN OBJECTIVES (TN 5-72-50)

Frank Teplitzky

The Instructional Development Control and Monitoring System (IDCMS) is a powerful and flexible tool that permits researchers to "random access" audio and/or video instructional segments for purposes of experimentally assessing and intensively analyzing the elements of these segments. IDCMS contributes both to the technology of instructional research and to the development of specific instructional programs. This document describes the foundation upon which IDCMS is built and the phases in which it will be developed. The phases are necessarily tentative since technological advancements and special user requirements will undoubtedly modify current developmental plans.

Evolution of system architecture. The hardware foundation for building SWRL IDCMS capability was configured by AMPEX Corporation to SWRL specifications prepared in consultation with ASCI of Palo Alto. This configuration, termed IDCMS Version I hardware, is shown in Figure 1.

An important consideration in selecting Version I hardware was that it be of modular design to avoid obsolesence and that it permit functional as well as modular growth of IDCMS. Each increment in capability with be achieved with configuration hardware, computer software, and terminal design enhancements to provide the functional characteristics necessary for a range of educational R&D applications.

SWRL

IDCMS
Version 1

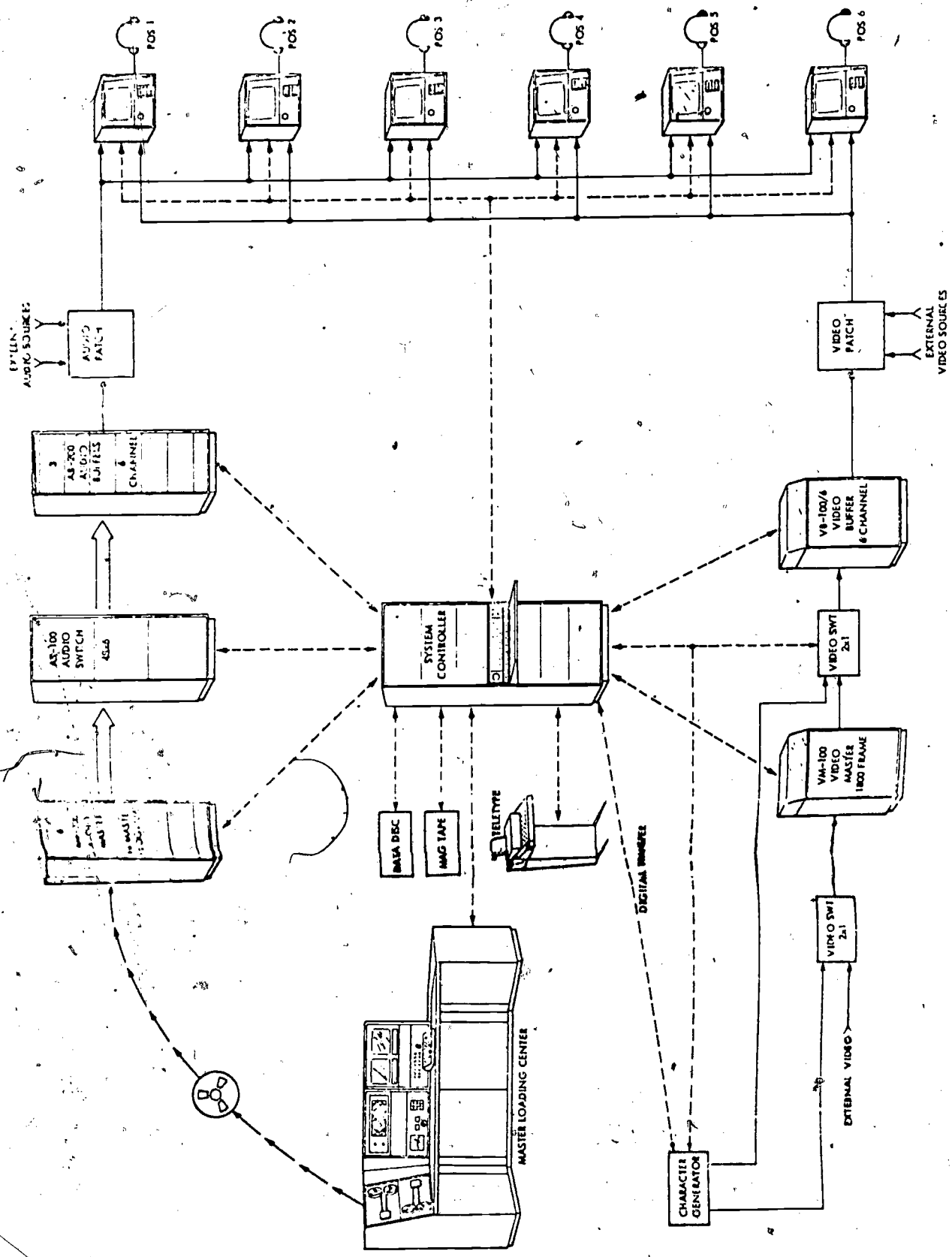


FIGURE 1

PHASE I

During Phase I, SWRL Computer Center staff will develop IDCMS operating procedures and skills necessary to work with IDCMS hardware and software components. A limited but powerful set of functions will be made available to Laboratory researchers so that moderately complex experiments can be run. Phase I operations will generally include:

1. Presentation of audio and visual stimuli to a user according to preprogrammed instructions stored in a digital system controller.
2. Mixing of video tape and character generator information for presentation on portable TV monitors at user terminals.
3. Presentation of pictorial and alphanumeric visual information, in color or black and white, on the TV monitors.
4. Control of the rate of information presentation by terminal operators.
5. Synchronization of visual and aural presentation sequences.
6. Immediate access to all analog information stored in the system's master storage components by all terminal users.
7. Recording of user responses entered via a portable keyboard for off-line analysis.

The basic components of IDCMS, as depicted in Figure 1, consist of a system controller, an audio subsystem, a video subsystem, and terminal stations. The subsystems are briefly described below.

SYSTEM CONTROLLER

The system controller is a NOVA 1200 Mini-computer, a general purpose digital computer processor manufactured by Data General Corporation. Its primary function in IDCMS is that of system controller of all IDCMS

elements. Each user terminal is interfaced to the controller, and the command and control linkage between the keyboard and all other IDCMS elements is through the controller. Linkage is established by either preprogrammed instructions stored in the audio buffer in digital code form, or by operator commands. For example, a request for a particular audio/video program is entered by pressing the required program identification numbers on the requestor's keyboard. This action transfers the request to the controller, which initiates duplication of the program at the user terminal. After transfer, the user keyboard is placed in a control mode to provide complete control of the audio buffer assigned to that keyboard. Each device in IDCMS has a control interface to the controller for status sensing and control functions. The control interface serves as the digital multiplexer for the controller input/output bus. The controller stores digital data on a digital magnetic tape drive and on a 2.5 million character digital disk memory. Special control interface units are provided to permit data transfer between these units and the controller. In addition, an interface to an alphanumeric character generator allows textual inputs to be superimposed upon video displays.

AUDIO SUBSYSTEM

A terminal user can select any one of the programs at any time, and be given near-instant access to the complete program, regardless of the number of other users already using the program. This enables each user to have complete control over his copy of the program.

Operation of the audio subsystem is based upon AMPEX high-speed magnetic tape duplication processes. Each terminal has an individual audio buffer control. Upon request of an audio program through the keyboard, the controller interprets the request so that the selected audio program is duplicated at high speed from a selected track on a master audio reproducer onto the user's buffer. Thereafter, this copy may be manipulated and controlled by the user into stop, start, rewind, fast-rewind, and fast forward operations. Duplication speed of the program is 150 inches per second. Tape reels can hold up to 25 minutes of program play time. It will be possible to store a maximum of 96 programs at any one time during Phase I operation.

Audio-Video Synchronization

Audio-video synchronization is accomplished by digital control codes imbedded in the audio programming; on play, the codes signal the controller to select and present the corresponding video frames.

In Phase I the coded address will automatically synchronize the audio tape with the video images regardless of user manipulation of the program. Since these are absolute digital codes, a user may rewind or advance his program and still maintain audio and video synchronization. Each program will be able to access a maximum of 128 pictures, which will be called-in sequentially as the audio is presented.

Audio Tape Preparation

Programming control codes has been made simple and convenient for the developer of instructional materials through the use of the Mastering

and Loading Center equipment. A portable AMPEX Model AG500 tape recorder is used to record experimenter-developed audio inputs onto 2-track source tapes. As the original audio recording is played back, the operator enters a code and advances to the next code. The codes are released by the controller as a series of modulated 55 hz bursts which represent a binary digital code. The 55 hz signal is recorded on the second track of the source tape. The source tape is then placed on-line and its codes are merged as a subcarrier with the audio information onto one of the eight tracks on a Master reproducer tape. The audio tape and video inputs may then be previewed at the Mastering and Loading Center on TV monitors. When a 55 hz signal is encountered during the play, the controller senses an interrupt and presents the next picture.

VIDEO SUBSYSTEM

Video access is based upon storage of hundreds of individually recorded video frames or images on the surface of a magnetic disk in the VM-100 video master disk unit. Upon command, a movable head selects the requested image from the master disk unit and the image is played back and duplicated on a fixed head buffering disk (video buffer disk unit) and reproduced on the student monitor as long as desired.

Integrated multi-media system capability is available in that the same monitors may be used for alphanumeric presentations, live camera, cable TV, broadcast material, and video tape presentation.

Program material is added to the master video storage disk using a video camera and a character generator. The master video disk has an

1,800 video frame capacity. Forty frames are used for test signals and indexing, and 1,760 are used for instructional programming. The character generator, manufactured by Telemation Corporation, can combine with video inputs to form a composite of alphanumeric and pictorial frames that can be transmitted to the station's video buffer and then to the user TV monitor.

Video System Operation

The programs developed by the experimenter can be audio, video, or a combination of both. Video-only programs are requested by the user in the manner similar to the request for audio/video programs. Pictures within the video program are available to the learner in sequence by pressing the "PIX Advance" or "PIX Reverse" buttons. Each time a selected "PIX" button is depressed, the picture sequences are advanced or reversed by one frame. By continuously stepping through the video disk using the "PIX Advance", the impression of slow motion may be obtained.

STATION CONFIGURATION

Each of the Learning Lab carrels in Version I consists of an audio/video terminal with a responder keyboard. The terminal's video component is designed around the SONY 12-inch Trinitron color television receiver; the audio component is from the random access program material, routed to a head phone set at the unit. To communicate with the system, each terminal has a portable 12-button access, control, and response keyboard. The keyboard resembles a touchtone telephone panel

except that, in addition to numerals, video and audio control functions are also identified (see Figure 2).

EXPERIMENT CONTROL

In an experiment with instructional segments (i.e., pretest, training, posttest, evaluation) where the segments to be administered require a decision, the experimenter will key in the identification of the next segment to be duplicated to his audio buffer. Segments will be played through to completion and the experimenter will control loops, skips, and extended alternative paths (see TN 1-72-01 for definitions).

PHASE II

Phase II general design goals are:

1. To provide random access video selection from any disk location. This will minimize the requirement to store duplicate copies for a program using a picture more than once.
2. To make available multi-program storage on audio master tape. Depending on the average length of an instructional segment, the audio master storage capacity will be significantly increased.
3. To develop an Experimental Control Language (ECL) to allow lesson developers to manage their programs more efficiently. ECL will be translated into Experiment Control Codes (ECC) by a batch mode computer translator program. These codes will reside in core memory and operate in synchronization with their audio program.

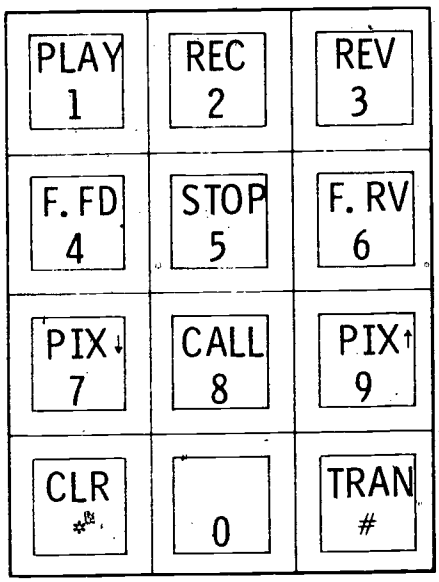


FIGURE 2

4. To expand the experiment control functions through newly defined EC codes. This is to include variable pacing, audio buffer control, etc.
5. To permit utilization of full alphanumeric keyboards at user stations.
6. To provide an automatic segment sequence control. This feature will allow the lesson developer to place an experiment control code in his instructional segment (or following it) to initiate the transfer of the next instructional segment to the user buffer automatically. This will automate interlesson branching and significantly reduce the overall lesson throughput time. The elimination of the requirement to manually key-in succeeding segments will also reduce keyboard input errors.

Even though most of the above items are computer software rather than hardware functions, another 8K core module must be added to the NOVA 1200 to permit programming of the functions.

PHASE III

By Phase III it is expected that Lab researchers will have had the necessary hands-on experience to contribute suggestions for making IDCMS easier to use and more appropriate to their needs. Toward this end, the following design objectives are planned:

1. The experimenter will be provided with an on-line interactive capability to permit him to manage his instructional sequences

even though others are using the hardware. Included will be an interactive program management capability which will allow him to:

a. enter a new ECL source program

b. edit programs to resequence the EC instruction set

c. save or delete program files

d. execute programs

Any combination of these operations can be performed at any station even though users at other stations may be performing other operations. In many respects IDCMS will resemble a mini-timesharing system, except that computer program development and system utility operations will need to be separately scheduled.

2. The implementation of "arithmetic operators" will provide a particularly powerful tool for the experiment designer. The operators will provide loop count control, student path tracing through a program, group path tracing, counter based automatic call-in of new program segments (computerized decision-making), and so forth.

3. Intra-audio buffer branching will be explored during this phase. The development of this capability will allow the computer to control audio buffer fast forward, reverse, etc. in a measured way.

Phase III enhancements to IDCMS capability will involve a combination of hardware modification, additional core memory, and skillful computer programming.

PHASE IV

The NOVA 1200 controller lacks the ability to perform complex on-line, real-time analyses of user response data primarily because it has insufficient capacity to accomplish analyses and system control simultaneously. A Phase IV design objective is therefore to extend IDCMS capability to the point where it can provide the experimenter with computational power beyond the limits of the NOVA 1200 mini-computer. To accomplish this, an on-line interface with either a large time-shared or a remote batch computer will be developed. This will allow the raw data of an experiment to be transmitted via telephone lines to the larger computer for detailed analysis. Once the data are processed, the reports generated, and the data bases are updated, the results will be transmitted back to the NOVA to be printed or used as criteria for computerized decision making.

ADDITIONAL IDCMS POTENTIAL

A potentially exciting aspect of the IDCMS developmental effort is in the area of input/output devices to broaden the range of response and feedback mechanisms. In TN 1-72-02, Follettie describes a variety of educationally useful response categories that can present a high order of flexibility for researchers. The use of speech comparators and analyzers, for example, are devices that offer great promise in the field of voice input/output. Tied to the Language Analysis Package and Music Analysis Package being developed at SWRL, computerized speech and music response evaluation may be investigated.

The concept of IDCMS is unique. Its development will be affected by Laboratory priorities and by technological advancements internal and external to SWRL. Full exploitation of IDCMS potential will require both resourceful computer system developers and imaginative and purposeful users.

Working Paper 5

PROGRAMMING SPECIFICATIONS FOR IDCMS, PHASE I (TN 2-73-10)

James M. Moloney

This document presents specifications for programs to be implemented for Phase I of IDCMS. The programs are categorized under the following four headings:

I. Systems Programs

II. User Programs

III. Utility Programs

IV. Diagnostic Programs

Systems programs are embellishments of the Data General Disk Operating System (DOS) architecture. They handle task priorities, establish job queues, handle interrupts and other system management functions.

User programs are designed to perform a specific task for the typical user. They may provide him with certain kinds of information processing and/or an expanded capability.

Utility programs provide a generalized data processing function. They facilitate day-to-day computer operation. They are not part of the DOS architecture and, hence, do not require a system generation for their implementation.

Diagnostic programs serve two functions. One function is to provide an analysis of particular instances of computer malfunction that will lead to an identification of the cause of the trouble. The second function is to provide statistics on different aspects of

system performance that indicate those parts of the system architecture in need of improvement.

This document is intended to describe the necessary software modification for Phase I of IDCMS; TN 5-72-50 is a minimal pre-requisite for understanding these modifications.

I. Systems Programs

1. Delete "delayed" state code
2. Constant time base
3. Keyboard software

II. User Programs

1. Response recording
2. Report generation
3. Character generator software

III. Utility Program

1. Copy/Compare Files

IV. Diagnostic Programs

1. System monitor
2. System log

I. System Programs

1. The Ampex supplied software contains code intended for their Pyramid system. Some of this code is not necessary for IDCMS operation. It includes the "delayed" state code and routines which are functional duplicates of routines in DOS (e.g., logical OR, ring buffering, etc.). Unnecessary code of this type will be deleted before incorporating IDCMS into DOS.

2. Several routines presently measure time using time bases, ranging from .5 sec to .001 sec increments. Such multiple

bases are justified for Ampex PYRAMID systems configured differently from IDCMS. In addition, it is necessary that all IDCMS routines operate on one fixed time base because response latency measurement will be required, thus, all IDCMS routines will be modified to run on a time base of .001 second.

3. Under current IDCMS software, the 12-button keyboard (see figure 1) may only be used to call up instructional segments and to initiate control functions. In Phase I, however, it will be necessary that the keys sometimes represent numerical responses rather than control functions. Thus, to the user, the system will appear to be in one of the three following modes of operation: (1) executive mode during which an instructional segment may be initialized and run; (2) function mode during which the keys provide control functions; and (3) response mode during which the keys represent numerical

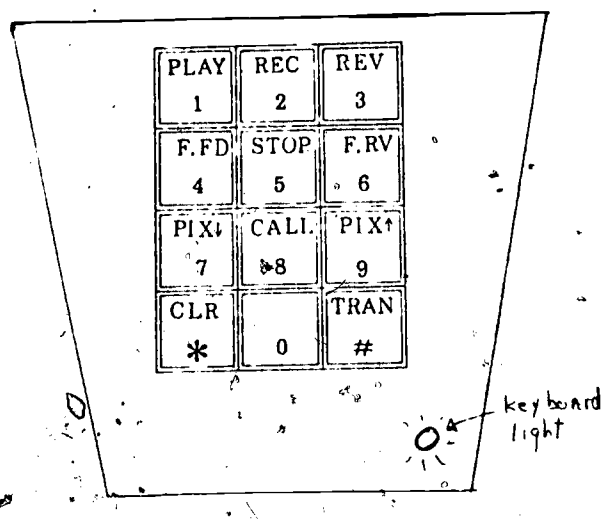


Figure 1.

1. PLAY - Play audio tape.

2. REC - If in "function" mode and tape is playing or stopped, change to "response" mode. The keyboard light will be illuminated when in "response" mode.

3. REV - Reverse audio tape at low speed.

4. F.FD - Fast forward for audio tape.

5. STOP - Stop audio tape.

6. F.RV - Fast reverse for audio tape.

7. PIX↓ - IF AUDIO TAPE IS STOPPED, causes previous picture in sequence to be displayed. Otherwise, it will stop tape and then change picture.

8. CALL - Causes the console teletype in the Computation Center to ring its bell and to print the student station number. CALL has no control function.

9. PIX↑ - IF AUDIO TAPE IS STOPPED, causes next picture in sequence to be displayed. Otherwise, it will stop tape and then change picture.

*. CLR - If in "function" mode, nonfunctional.
If in "response" mode, change to "control" mode.

0. - No control function.

#. TRAN - If in "function" mode, terminates session. If in "response" mode, acts as a robot.

responses. The following paragraphs specify the changes to be made in the keyboard software. Any key not specifically mentioned for function mode, retains its original function.

In executive mode, the TRAN key (#) will always signify that a user is waiting to log onto the system. To log on the TRAN will be followed by an experiment number nnnnn, and CLR. The experiment number will uniquely specify one instructional segment. This procedure will cause the system to start duplication of the

master tape into the station buffer. At the same time the system will accept a user number, nnnnn. The user number will consist of the user's CRC (1st 3 digits) plus two digits which identify the student/user.

This user number will be terminated by a CLR and, when the tape has been duplicated, the instructional segment will begin in function mode. At any point in the log on procedure before the tape has begun playing, the user may start over by pressing TRAN. Likewise, as soon as the user number is terminated, the system is effectively in function mode, and the corresponding function of any key pressed will be carried out as soon as possible. Upon entry into function mode, the tape will be in the STOP state.

The functions of five keys are to be changed for function mode. The REC key will change the mode to response mode if the audio tape is playing or stopped. The PIX↓ and PIX↑ will stop the tape if it is moving and then change the video frame backward (PIX↓) or forward (PIX↑). The CLR key will have no function and the TRAN key will return the user to executive mode and a log on will be expected by the IDCMS monitor.

In response mode, all keys except CLR are to be considered as responses. The CLR key will change the mode to function mode. The TRAN key will be used as a single character rub-out. For example, if 1234##56 is typed, then the 3 and 4 are considered to be rubbed out and the response is taken to be 1256.

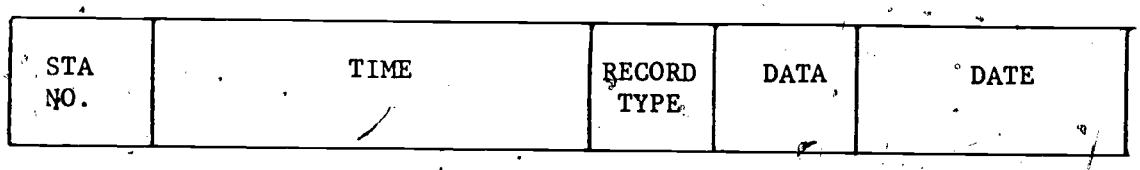
Note that in this mode the TRAN key will not return the user to executive mode.

II. User Programs

There are three user programs to be specified for Phase I. First, a data recording capability will be added. The second program, a report generation program written in Fortran, will provide the user with a complete subject protocol for any subject run on IDCMS. The third program will provide the initial software for an expanded character generator capability in the later phases.

1. The general data recording procedure will be as follows. Each unit of data will consist of four sixteen-bit words. At certain specified times which will be precisely defined below, a data record will be written into a buffer in core. When full, the contents of the buffer will be transferred to the disk and finally to a seven-track magnetic tape.

The data record will have the following format:



The station number (STA NO.) is a four-bit binary number between one and six. It will serve to identify the record for later processing.

The time is a 28-bit byte containing the elapsed time measured from midnight each day.

The record type is an 8-bit binary number which identifies the type data that the record contains. There are presently only the following five record types defined:

16 - Picture code
17 - Time of display
32 - Keyboard stroke
33 - Header information
48 - System message

Multiples of 16 were chosen for the primary record type identifiers to allow for two levels of associative information for future classification and analysis. For example, the 8-bit binary code for a picture code is $0001\ 0000_2$. A record will be written using this code each time a picture code is encountered. It is also useful and interesting to know when the picture was actually displayed on the screen since there is a delay between encountering the picture code and displaying the picture. The time of display (TOD) code is defined as $0001\ 0001_2$ (17_{10}). Then a TOD record is written given the exact time a picture is displayed.

The next 8 bits of the record contain the data. There will be as many data types as there are record types. The data for a picture code (16) will be the binary picture number. Since a TOD code (17) will always follow a picture code, the data field for a TOD may be null. The data for keyboard stroke (32) will be an ASCII character representing the key pressed. For digits, this is straightforward. For the functions, unique encodings must be specified. When assigning these codes, the conversion to full

alphanumeric keyboards should be kept in mind. The header information (33) will be the same as type 32. The change in record type is to allow the report generation program to more easily recognize sign-on information. The data for a system message (48) will be the binary message number.

The final 16 bits of the record will contain the date. The date will be a binary number representing the day, month and year. Day zero will be January 1, 1900.

During Phase I, the DOS magnetic tape software will be used. The DOS User's Manual makes the following statement about 7-track tape units.

> Data recorded on 7-track units is necessarily encoded. This is accomplished in the following manner;.
>
> Original Data Word

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

> Encoded Data Words

x	x	x	x	0	1	2	3	x	x	x	x	4	5	6	7
x	x	x	x	8	9	10	11	x	x	x	x	12	13	14	15

> Every data word written on a 7-track unit as two encoded data words. When actually written on the tape the data will look as follows:

x	x	0	1	2	3
x	x	4	5	6	7
x	x	8	9	10	11
x	x	12	13	14	15

2. The report generation program will accept as input the response data tape described above. Figure 2 shows each type of record

STA NO.	TIME OF DAY	$0001\ 0000_2$	BINARY NO.	DATE

a.) Picture code record

STA NO.	TIME OF DAY	$0001\ 0001_2$	BINARY NO.	DATE

b.) Time of display record

STA NO.	TIME OF DAY	$0010\ 0000_2$	ASCII CHAR.	DATE

c.) Keyboard stroke record

STA NO.	TIME OF DAY	$0010\ 0001_2$	ASCII CHAR.	DATE

d.) Header information record

STA NO.	TIME OF DAY	$0011\ 0000_2$	BINARY NO.	DATE

e.) System message record.

Figure 2.

SAMPLE FORMAT

EXP:DDDDD	USER:DDDDD	STA:D	DATE MM/DD/YR	(START TIME) HH:MM(1)

* *

EVENT	TIME	DATA	LATENCY
KEY	3:30:04.713	REC (2)	
PIC	3:30:05.002	1	
TOD	3:30:06.999 (3)		
KEY	3:30:07.454	3	2.452 (4)
KEY	3:30:07.678	4	0.224
KEY	3:30:07.890	5	0.212
PIC	3:30:10.225	2	
TOD	3:30:11.019		
KEY	3:30:14.329	6	4.104
KEY	3:30:14.489	1	0.160
.	.	.	.
.	.	.	.
.	.	.	.
SYS	3:33:46.321	M10 (5)	
.	.	.	.
.	.	.	.
.	.	.	.
PIC	3:40:34.747	17	
TOD	3:40:35.123		
KEY	3:40:40.333	0	5.586
KEY	3:40:41.586	CLR (6)	
KEY	3:40:41.937	F.FD.	
KEY	3:40:41.123	STOP	
KEY	3:40:42.456	PLAY	
KEY	3:40:42.678	REC	
PIC	3:40:45.345	21	
TOD	3:40:47.890		
KEY	3:40:49.567	8	4.222
KEY	3:40:49.999	3	0.432

* * * END OF EXPERIMENT DDDDD * * *

M10: POOR PICTURE QUALITY

Figure 3

that the program must handle. It will provide as output a report similar in form to Figure 3, except for the parenthesized expressions which are used solely for descriptive purposes in this document. Latency will be computed for response mode data only.

The header information (1) is obtained from a string of records of type 33 (Figure 2,d). The first record will contain a TRAN in the data field and the start time, date and station number are obtained from this record. The encountering of a TRAN for any station will always signify a new sign-on and the generation of a new report. The TRAN is followed by a sequence of type 33 records containing ASCII digits in the data field followed by a type 33 record with CLR in the data field. The experiment identification number is the concatenation of the ASCII digits.

The CLR is followed by another sequence of type 33 records containing ASCII digits and terminated by another CLR. These digits are the user number. No further type 33 records will be found.

In the event that a type 33 record containing a TRAN is followed by the one or more TRAN records, the last such record is to be used for the header information. A single isolated type 33 record with a TRAN in the data field is ignored.

The first event on the sample report (2) indicates that the user requested response mode at 4.713 sec after sign-on (assuming that sign-on was at 3:30). The information was obtained from a type 32 record with the ASCII representation or REC in the data field. The time is the value of the time of day field for this record.

The next event (3) shows the time that a picture was actually displayed. This information always follows a picture code and is contained in a type 33 record.

The next event (4) is the occurrence of a picture code (PIC) at 5.0002 after sign-on (assuming start time is 3:30:0.000). The information was obtained from a type 16 record with the picture number stored in the data field in binary. A picture code sometimes indicates that a keyboard response is expected. In such cases, the above-mentioned type 17 record will be followed by a sequence of one or more type 32 records containing ASCII digits in their data field. These records provide the information for printing the three lines following (4). The latency to first response (in msec) is obtained by subtracting the time of the start of the immediately preceding TOD. The latency to each additional keystroke is computed by subtracting the time of the keystroke from the time of its immediate predecessor. If a type 17 record is followed by anything but a type 32 with ASCII digits in its data field, then only the event, time and data information must be printed (6).

PICTURE DISPLAY TIME REPORT DATE

EXPER NO.	PIC CODE	AVE DISP TIME
1	1	447
1	2	798
1	3	1096
1	4	582
.	.	.
.	.	.
.	.	.
1	17	693
2	1	891
.	.	.
.	.	.
.	.	.
92	121	924

OVERALL MEAN DISPLAY TIME = 777 millisec.

Figure 4.

Occasionally, type 48 records will be encountered. These will give rise to entries such as (5) in figure 3. The error message will be decoded at the bottom of the page as shown.

It is imperative that the program be written in a highly modular fashion since it is likely that new record types may be added and other report formats may be requested. For example, suppose it is desirable to obtain data on the delay in actually displaying a picture after a picture code is encountered. A report generated for this purpose, using information contained in a type 17 (TOD) record, might be of the form shown in Figure 3.

The ASCII codes for the control functions and the special characters will be provided by the systems programmer. A table of system messages will also be developed.

3. The third user program involves designing a disk stored "message" file for input to the character generator and developing the routines necessary to associate the message with related program video images. In addition, character generator software will be expanded to allow input to the disk message file from the keyboard. Investigation will continue on software implementation of such character generator functions as centering on the page, snake up, snake down, etc.

The disk storage of messages for input to the character generator should be designed to conserve disk space as much

as possible. A storage scheme developed by Porch and Ambler will be outlined here. This scheme is one of many that may be employed for message storage.

There will be the following storage components:

(1) A message directory with pointers to the first word of each successive message stored in the main storage area.

(2) The main storage area contains messages.

(3) Messages are made up of elements followed by a message terminator.

(4) An element is a string of ASCII characters or a call to another message.

(5) A call consists of a special function character followed by a message number.

The following example is not intended to specify the exact internal storage technique, but merely to serve as an illustration. Note the small expansion in storage needed to save input message No. 2.

EXAMPLE

Input Message No. 1:

THIS MESSAGE SHOWS THE MESSAGE STORAGE METHOD

Storage Illustration:

1. THIS (2) SHOWS (3)
2. MESSAGE
3. THE (2) (4) METHOD
4. STORAGE

Input Message No. 2:

THE MESSAGE STORAGE METHOD SAVES STORAGE SPACE

Additional Storage:

5. (3) SAVES (4) SPACE

Calls to messages will allow significant reduction in disk storage requirements since each element need only be stored once, regardless of the number of messages in which it is included. It is possible that the ECL source code may be stored in an analogous manner for Phase II.

III. Utility Programs

The programs described in this section are of general use to system operator and programmers in the daily operation of the system. They will provide several necessary information processing functions. Although they will seldom, if ever, be used by the general user, they are not systems programs and will not change the overall system architecture.

The utility program discussed here is a magnetic file copy and/or compare program. This program will perform three separate functions. It will be able to move data from one specified file to another. It will be able to compare data from one specified file with another. During the comparison, the program will provide detailed error messages, if necessary, and then continue until end of tape or file.

IV. Diagnostic Programs

The major diagnostic implementation method will be the use of the conditional assembly feature of the NOVA assembler. By using this facility, it will be possible to insert debugging code throughout the system but not assemble this code under normal operating conditions. The special code may be activated when necessary by a simple one-step re-assembly.

1. There will be a routine to monitor and evaluate system timing conditions. This routine will use the starting address of the interrupt service routine as an access point. For each interrupt (including the clock) a counter appropriate to the interrupting device is incremented, then control is returned to the system.

   Two types of critical timing information will be made available from utilization of this routine: (1) How many different types of interrupts occur per unit time. (2) How critical is the latency between the interrupt for each device and its handling response.

   The first type of information may be obtained by monitoring system interrupts with the routine for N seconds. The second type of information may be obtained by causing successively greater delays before handing process control back to the system. When the system stops acknowledging interrupts, the critical timing latency has been reached for that device.

2. A routine will be written to keep a simple log of the day's events. This log will be kept as a disk file. Such a routine will provide information on utilization, system load, sequence of significant events, etc., when problems occur and, thus, aid in the analysis of these problems. Information included in the Phase I log will be such things as:

Time: Terminal N is starting lesson M
Time: Terminal N is down
Time: System crash
Time: Magnetic tape write error
etc.

Working Paper 6

CONTINGENT INSTRUCTIONAL ADVANCE: IMPLICATIONS FOR IDCMS (TN 1-72-05)

Joseph F. Follettie

Student-system interactive instruction or research requires decisions to advance S or to loop off the mainline contingent upon some or all of the information reflected in prior performance. The SWRL Instructional Development Control and Monitoring System (IDCMS) in time will be appreciably used to establish characteristics of effective-efficient interactive instruction. The design of IDCMS in Version 1 hardware configuration now is appreciably completed. The design was premised on a general view concerning what the system will be asked to do and cost considerations. Developing views on how the system in Version 1 form desirably will be used now make entertainable the proposition that slight modifications in hardware design may be warranted. Although perhaps a less imperative matter, parallel comments seem applicable to software to be provided by the contractor. The particular slight modifications in the contracted system that may be warranted can only be identified and evaluated in consequence of increasing specificity regarding desired general system functions. Among system functions, the interactive one presently invites closest scrutiny; many research functions are subsumed by the system's interactive function. The scenario to be presented illustrates an interactive structure in sufficient detail to permit qualified staff to evaluate the contracted IDCMS configuration against functions inherent in the interactive structure. Such an approach is only as useful as its illustrative argument is compelling. Hence, the possible executive decisions that system "deficiencies," so defined, invite are: a) If costs are acceptable, modify contracted design for the system to achieve illustrative functions. b) Modify the interactive requirement to correspond to the contracted system. A compromise decision falling between these alternatives also is possible.

A mainline instructional program is one that S will negotiate to a program-defined exit if the instructional control decision at each decision point falling along the mainline is positive (+). That is, if at each point tested S reveals a criterion level of proficiency for the program skill(s) (PS) tested, then he will advance along the mainline to a predetermined succeeding lesson until, at last, he exits through a terminal decision point for the program. However, whenever his performance warrants a negative (-) instructional control decision, S will loop off of the mainline. That is, S will loop off of the mainline wherever a test reveals achievement of less than a criterion level of proficiency for tested program skill(s).

If a test warrants a negative instructional control decision--that is, reveals subcriterion proficiency--then two major causes for this decision can be discerned: a) prior relevant mainline instruction or the conditions of its administration--e.g., pacing--are ineffective for this particular S or b) prerequisite skills not taught on the

mainline are deficient. Two sorts of prerequisite skills--distinguished on the basis of the point in the instructional progression wherein their proficient employment is required--can be discerned: a) entry skills (ES), whose proficient employment is required during earliest lessons of the instructional program and b) enroute prerequisite skills (MS), whose proficient employment is required at later intermediate points in the instructional sequence. The point at which such skills become relevant would not be particularly important as a basis for distinguishing the two sorts of prerequisite skills if skills of both sorts became relevant one at a time. However, it tends to be true that several prerequisite skills become relevant at the outset of instruction, whereas prerequisite skills that become relevant later in instruction do so one at a time. Illustrative interactive instruction initially will make use of this difference in temporal concentration of entry and enroute prerequisite skills. However, later remarks will place ES and MS in the same set (EMS).

A fundamental assumption underlying the structure to be illustrated is that student-system interaction should reference to moderately extensive instruction under the condition of multiple usership (e.g., 6 Ss) Some of the problems we will encounter disappear or are considerably lessened if instructional extent is taken as one-third of what we will show. When elsewhere we examine time-pressured retrieval-transmission from stores containing much larger numbers of "entities," such a lessening of extent will bound the effort.

Moderately extensive instruction is here normatively defined as 15 30-minute lessons, such that an S whose rate of acquisition (referenced to the illustrative instruction) is average will complete the program by working 25 minutes per day on normative instruction and supplementing instruction as needed. Figure 1 provides a diagrammatic view of the illustrative instruction. Open circles reflect mainline instruction; closed circles, the (identified) prerequisite skills. (A foreseeable outcome of interactive research is that it will invite +H hypotheses concerning relevance of formerly identified prerequisite skills.)

Level 1 lessons of Figure 1 may be interpreted either as addressing single program skills without subordinate skills structure or as addressing a program skill that is superordinate to a set of subskills. The former view yields the simplest possible illustrative program. Since the program already has been extensively oversimplified, the second interpretation is used here: We characterize Level 1 lessons of Figure 1 as reflecting a program skill that integrates two subskills previously taught in the same lesson.[1]

---

[1]Particularly where rule learning and generalization are required--as in phonics instruction--Figure 1 dramatically oversimplifies the situation by failing to reflect buildup of the rule set over lessons.

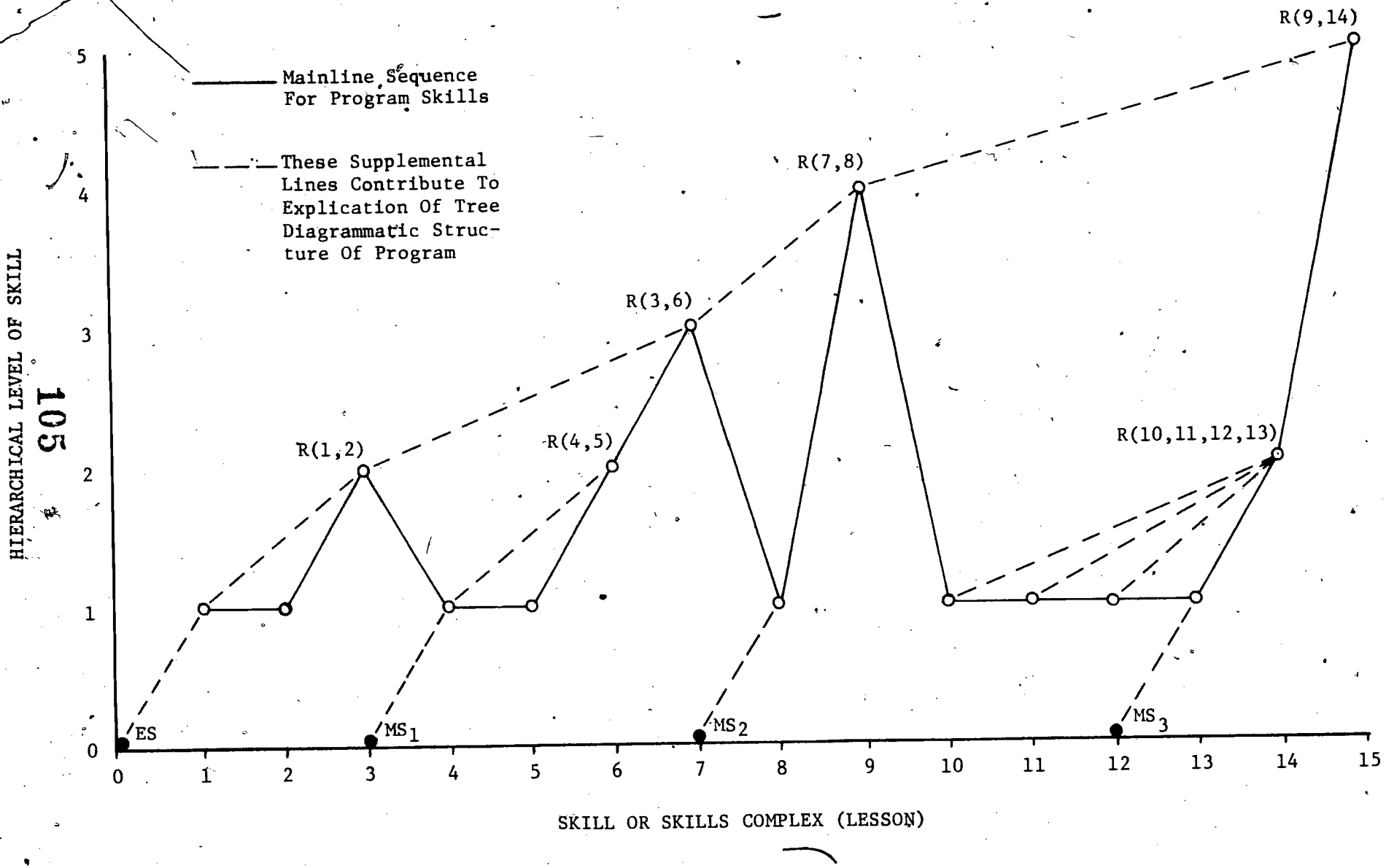


Figure 1. An Illustrative 15-Lesson Program, Showing Sequence, Skills - Integrative Structure (R = Integration of Skills), Entry Skills (ES), and Enroute Prerequisite Skills.

Entry Skills (ES)

Any instructional program is predicated on one or more entry skills. These skills are required during earliest lessons of the instructional program. However, they are taught only if evaluation reveals that S performs below criterion proficiency level for them. It is typical to class as entry skills only those skills for which it can be assumed that at least half of the entering Ss will be criterion proficient when tested at the outset of instruction. Earliest lessons of given instruction typically will require S to be criterion proficient in several such skills. Because so many entry skills require evaluation at the outset of instruction, we assume that an initial test, consisting of a perfunctory subtest for each entry skill, will either clear S for initial instruction or will yield one or more hypotheses to the effect that E may be deficient in one or more entry skills. Should S be cleared for mainline instruction (+ES), tenability of one or more hypotheses of form H:-ES_i still might be established later in consequence of diagnostic characteristics of the evaluation system. Should a hypothesis H:-ES_i be entertained in consequence of administering the perfunctory entry skills test, then deficiency in ES_i might be further evaluated using a more extensive test EV_1-ES_i (see Figure 2).

We assume that skills analysis referencing to given instruction should be extended downward no further than one level below the level of the entry skill. In consequence, if H:-ES_i is found tenable, then one level of formal instruction culminating at the ES_i level of a proficiency hierarchy can be designed. We denote such instruction IN-ES_i. Effectiveness of such instruction can be tested using a second version of the test for criterion proficiency in ES_i--denoted EV_2-ES_i. Should this test also reveal tenability of H:-ES_i, then one's options would be either to administer versions of IN-ES_i and EV-ES_i as required to raise S to ES_i or to administer informal instruction and conduct informal evaluation culminating in the decision +ES_i. We show the second of these options in Figure 2. Which option would prove most apt probably will depend on the particular characteristics of ES_i and, in research setting, of E and of the system that supports student-system interaction.

Where this is allowed, various system-burdening requirements emerge. First, the system must store a rather large number of novel rule words that permit testing for rule mastery in all applicable intraword positions and in all applicable word contexts. Second, so many alternate diagnostic tests can be imagined that on-line composition of tests becomes required, which may entail complex coding of word items. Were we to admit this complication into the illustration, then the lengthy paper that lies beyond would not be required. The system simply could not handle extensive rule learning and generalization within the context of moderately extensive instruction.

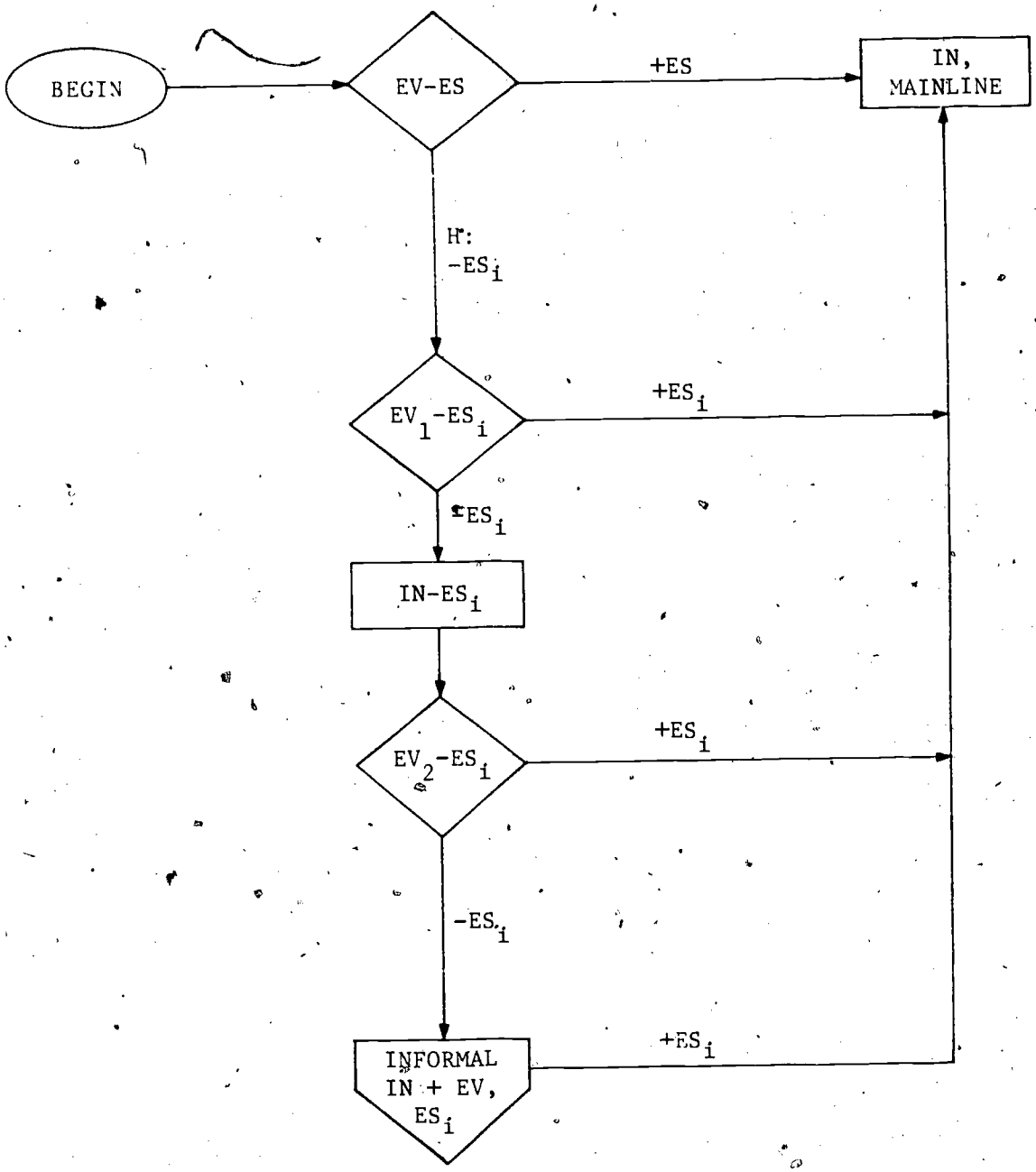


Figure 2. Illustrative Flowchart for Entry Skills (ES) And An ith Entry Skill (ES_i) In Particular For Which S Is Unacceptably Proficient, Treated Interactively. H = Hypothesis, EV = Evaluation, IN = Instruction, Minus Sign = Unacceptable Proficiency Level, Plus Sign = Criterion Proficiency

Enroute Prerequisite Skills (MS)

Enroute prerequisite skills differ from entry skills only with regard to the point in instruction wherein they become germane. These skills--denoted MS--become prerequisite to instruction at intermediate points during instruction, rather than during earliest lessons. Typically, such prerequisite skills will become germane one at a time. That is, unlike entry skills--several of which will prove germane at or very nearly at the outset of instruction--enroute prerequisite skills will warrant consideration singly at different points in instruction. In consequence, it is not necessary to employ an overall perfunctory test--EV-MS--for these skills. Rather, at the point wherein a given such skill MS_i becomes germane, a definitive test EV_1-MS_i may be administered (see Figure 3). Again, one's options are a series of versions of IN-MS_i and EV-MS_i which formally instruct and evaluate S until the condition $+MS_i$ is reached or informal instruction and evaluation culminating in $+MS_i$ after preliminary formal instruction and evaluation. Again, we show the second option in the flowchart.

As with entry skills, categorization of S as $+MS_i$ need not be irrevocable. If later evaluation is made suitably diagnostic, then it remains possible at some later point in instruction again to entertain H:-MS_i.

Program Skills (PS)

The instructional design-development effort assumes that not all Ss will require administration of instruction addressing ES and MS; however, it is assumed that all Ss will require administration of every program skill.[2]

Figure 4--an oversimplification to be corrected in later remarks--begins with first instruction for some program skill PS_i falling on the mainline; this instruction is denoted IN_1-PS_i. The illustration assumes entertainability of three hypotheses: a) HO:+PS_i, b) H_1:-PS_i with cause -MS_i, c) H_2:-PS_i with cause ineffective IN_1-PS_i (ineffectiveness defined on S). In consequence of S's EV_1-PS_i performance, we either will accept H_0--or, for the benefit of those who do not accept hypotheses, strongly entertain it--or will entertain H_1 or H_2. If H_1 or H_2 exhaust the possible causes of -PS_i, then it is not necessary that EV_1-PS_i provide a definitive basis for choosing between H_1 and H_2 (although

[2]Should empirical effort reveal this assumption not to be tenable, then instructional redesign and redevelopment would be required consonant with the proposition that all Ss will require administration of every program skill.

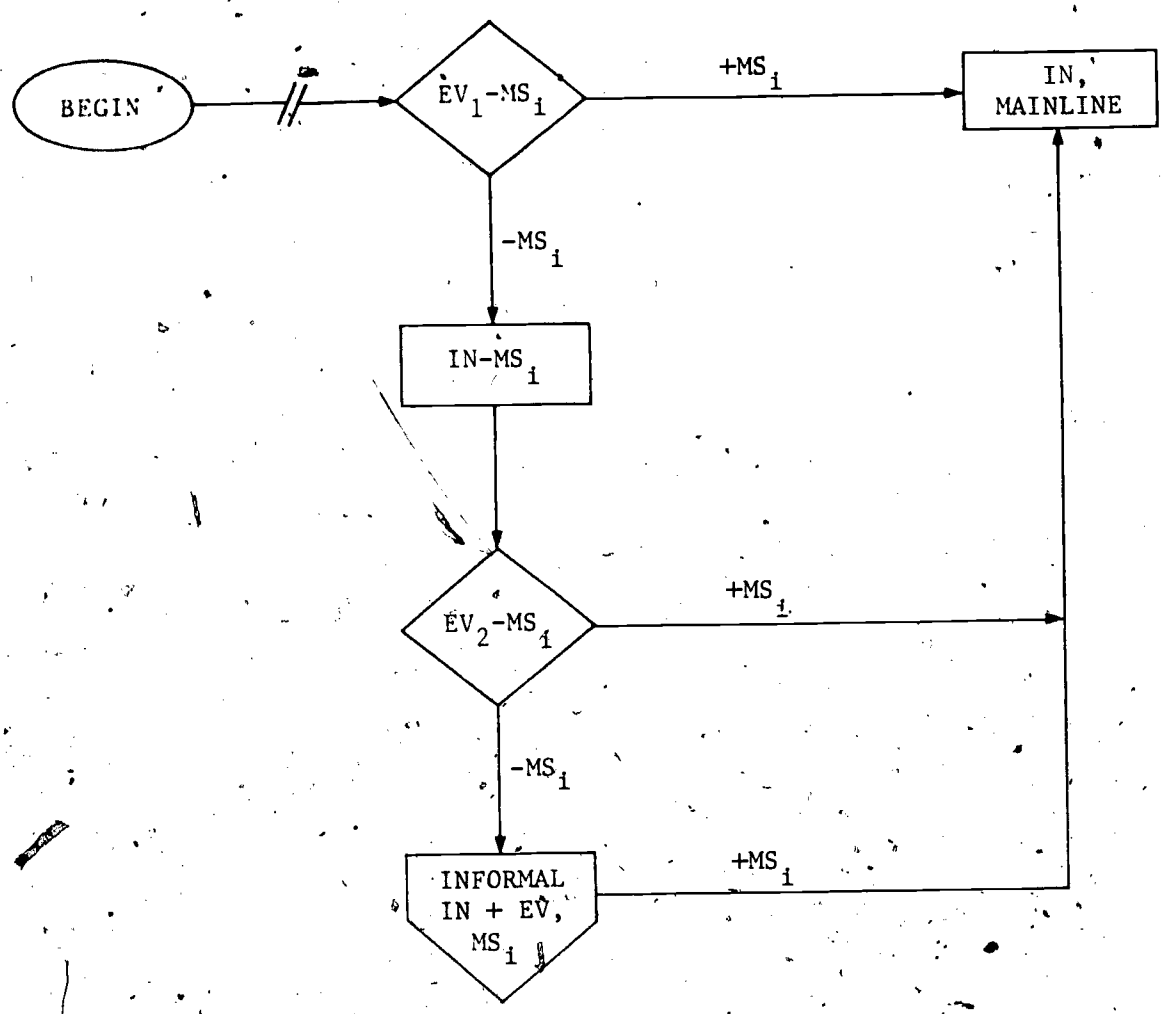


Figure 3. Illustrative Flowchart For an ith Enroute Prerequisite Skill (MS_i) Treated Interactively. EV = Evaluation, IN = Instruction, Minus Sign = Unacceptable Proficiency Level, Plus Sign = Criterion Proficiency.

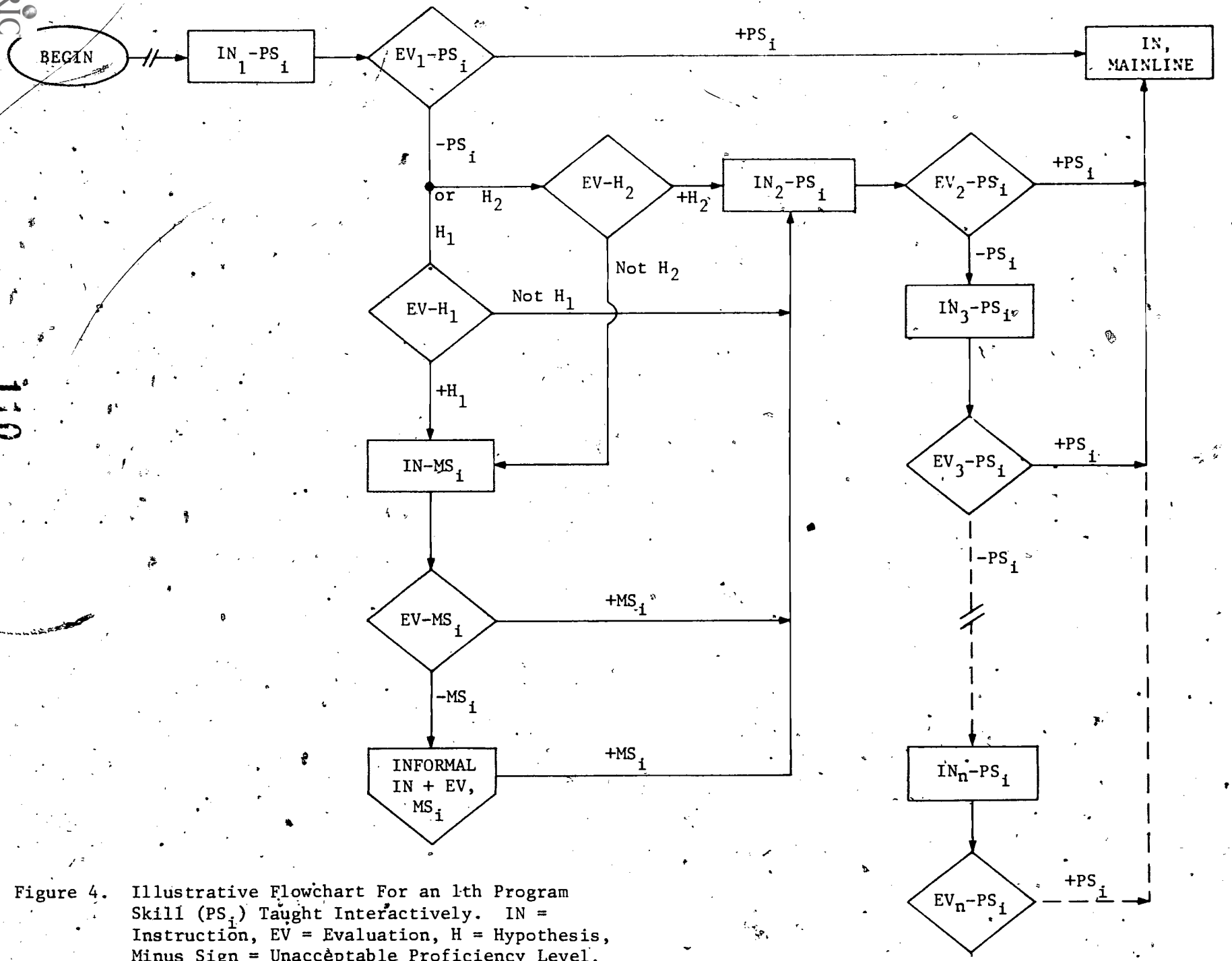


Figure 4. Illustrative Flowchart For an ith Program Skill (PS_i) Taught Interactively. IN = Instruction, EV = Evaluation, H = Hypothesis, Minus Sign = Unacceptable Proficiency Level, Plus Sign = Criterion Proficiency Or H Is Acceptable.

instructional efficiency will be higher if the test is definitive regarding cause). Figure 4 has us evaluate either H; if it is rejected, then the other H is taken as tenable and prescriptive instruction is administered accordingly. If H_2 is found tenable, then one alternative is to treat PS_i in greater instructional depth (or to elaborate on it, to use alternate exemplars, to make its logic more explicit, etc.) An extended series--IN_2-PS_i + EV_2-PS_i through IN_n-PS_i + EV_n-PS_i--might then be devised to take instruction to whatever level of elaboration experience proves useful. Alternatively, at some point early in such a series, one might switch over to informal instruction and evaluation. While Figure 4 terminates the prescriptive series that is consonant with accepting H_2 on formal instruction and evaluation, it often will be true that an acquisition problem can only finally be overcome in consequence of informal intervention by E. Such an exit from prescriptive instruction is uncritical here because it will not be counted in the count of program elements that burdens the system.

A Closer Look at Prerequisite Skills

The distinction between ES and MS probably can be defended on grounds of evaluative efficiency when instruction is extended. However, the present illustration probably will not suffer if we collapse the two--denoted collectively EMS. Let EMS = 8. We will assume that it makes sense to test any H_1 (that a prerequisite skill is deficient) using a cursory screening test EV-EMS. If, as we will, we then assume that it is necessary to follow up on the evidence provided using a restricted test $EV\text{-}EMS_i$, we are assuming in effect that the diagnosis based on the cursory test may be in error. I believe such an outcome should throw S back to a second version of EV-EMS. Let us assume that the system stores four such versions EV_1-EMS through EV_4-EMS. These are stored as separate, randomly accessible segments because if retrieved as a sequence we will need tell the system where in the sequence we wish it to transmit.

Since EMS = 8, we require 8 $EV_1\text{-}EMS_i$ and 8 $EV_2\text{-}EMS_i$. Letting each of these constitute one segment, then 18 such evaluative segments must be stored. Consonant with earlier remarks restricting formal instruction of defective prerequisite skills to one treatment, which we interpret as two segments, storage of an additional 18 segments will be required. The prerequisite skills storage, in segments, then is 36--4 EV-EMS, 16 $EV\text{-}EMS_i$, and 16 $IN\text{-}EMS_i$.

If EV-EMS yields $+H_1$ (or EV-H_2 yields not $-H_2$) then it will be required that one of eight 4-segment sequences be retrieved and presentation to S be initiated. Such sequences have the form $EV_1\text{-}EMS_i$ + $IN\text{-}EMS_{ia}$ + $IN\text{-}EMS_{ib}$ + $EV_2\text{-}EMS_i$. While S may exit from the sequence following EV_1 or EV_2, if he does not exit at EV_1, then he will continue through the sequence in fixed order. Hence, the illustrative instructional program

requires that four randomly accessible one-segment program elements and eight randomly accessible 4-segment program elements be stored for purposes of evaluating and instructing prerequisite skills.[3]

Segment Characteristics

For present purposes, we will define segments on time. Let the segment consist of 2-20 items, where an item may be audio only, video only, or audio + video as such characteristics can be discerned by an uncritical review of audio program and video frame storage for IDCMS. (We say uncritical because number of storable audio programs may be at issue.) Let the segment be two minutes long on the average, with number of items--2, 20, or an intermediate number-determined by mean item duration. What will vary is duration of the audio message. We assume that every audio message space will be preceded by an 11-bit digital code[4] whose average duration will be just in excess of 1.5 seconds and whose average separation from the audio message space--measured from the end of the code to the beginning of the message space--will be 1 second. We assume further that a stop interval lying beyond the audio message space will use 1.5 seconds of tape. Thus, if the segment contains 20 items, then an item will occupy 6 seconds of audio tape and the audio message space can only be 1.5 seconds long (because code, code separation, and stop intervals use 4.5 seconds). On the other hand, a 2-item segment on the average will feature 55 second audio messages. Wherever one instructional sequence (without testing interruption) features a series of instructional segments, then the instructional sequence, together with the test that terminates it, can be stored as a sequenced program element. Program element storage for prerequisite skills is four 2-minute program elements and eight 8-minute program elements--or 12 elements, 72 minutes.

---

[3]While it is likely that program elements referencing to higher-numbered prerequisite skills will not need be stored throughout instruction, a continuing requirement to store 12 elements presently appears sufficiently slight that it would not be worth the trouble to simulate the slightly lower requirement that would characterize a best day requirement for six Ss. Worst day would require that all elements be stored.

[4]The system as contracted contemplates using 7-bit codes. These suffice to random access to 128-track files--which requires only four bits. If seven bits really are required just to do this, then perhaps we require 14 bits rather than 11 to be able to get to any track number of video storage. While we do not require random accessing to individual tracks for purposes of serving illustrative instruction, the requirement will go beyond a capability for accessing to the starting point of any of just 16 128-track video files. Rather, we may need to be able to access to as many such files as we define program elements.

A Closer Look at Program Skills

Foregoing remarks have dispensed with H_1. H_0 accepted, S continues on the mainline. H_0 rejected and H_1 rejected, H_2 must be accepted (unless one hypothesizes some previously-unidentified prerequisite skill--the signal to start over). Just what H_2 entails turns upon lesson level in the skills hierarchy (see Figure 1). Level 1 skills complexes are independent of other skills complexes. If H_1 is rejected following rejection of H_0, then subcriterion performance detected by a Level 1 test limits the search for defective instruction to that for the particular lesson in which the Level 1 test occurs. Let us look at the Level 1 mainline to clarify where one can go in the event that a Level 1 test detects subcriterion performance.

We have asserted that every Level 1 lesson (Figure 1 shows nine) will feature two subskills that are taught and then integrated. The integrated program skill then has the subscript of the lesson's number. Each mainline treatment of a Level 1 lesson then will feature three IN-EV sequences. For present purposes we assume that any such sequence consists of two instructional segments and one evaluative segment. Sequences, or program elements, for a Level 1 lesson are:

1. $IN_1\text{-}PS_{i1a} + IN_1\text{-}PS_{i1b} + EV_1\text{-}PS_{i1}$

2. $IN_1\text{-}PS_{i2a} + IN_1\text{-}PS_{i2b} + EV_1\text{-}PS_{i2}$

3. $IN_1\text{-}PS_{ia} + IN_1\text{-}PS_{ib} + EV_1\text{-}PS_{i}$

Each sequence terminates on an EV which, detecting subcriterion performance, will jerk S off the mainline. Hence, it makes sense to view Level 1 mainline storage as 9 (lessons) x 3 (sequences) 3-segment program elements, or 27 6-minute program elements.

If H_0 and H_1 are rejected in consequence of administration of $EV\text{-}PS_{i1}$ or $EV\text{-}PS_{i2}$, then there seems just one place to go--to an alternative version of $IN\text{-}PS_{i1}$ or $IN\text{-}PS_{i2}$ (or to an alternative version, with alternative procedure--e.g., regarding pacing). Were we to follow Figure 4 literally, then we could overload any system at this point, simply by requiring many versions of each sequence. For present purposes, let us require just one alternative version per mainline instructional sequence at Level 1.

If H_0 and H_1 are rejected in consequence of administration of $EV\text{-}PS_{i}$, then the difficulty could reside at any of three addresses. Either instruction is ineffective for one of the two subskills or it is ineffective for integration of these subskills. While this could necessitate a longer-than-one-segment $EV_1\text{-}PS_{i}$, its follow-on implications for alternative sequences need not differ from those for subskills. That is, again we will require just one alternative version per mainline instructional sequence that integrates subskills at Level 1. However,

if the difficulty is identified as being at the subskill level and an appropriate alternative sequence is negotiated, then we wouldn't wish to have S negotiate new instruction at the integrative level just to be able to negotiate a new version of EV-PS_i. Hence, let us require a single additional version of EV-PS_i, to be used just to evaluate the program skill at the integrative level. In consequence, Level 1 looping to alternative versions of mainline instruction will require (if we circumvent EV-H_2 testing) 27 3-segment and nine 1-segment program elements. This amounts to 27 6-minute and nine 2-minute elements.

H_0 and H_1 rejected at higher levels, the source of difficulty is a more complex matter. Lessons 3, 6, and 14 of Figure 1 are at Level 2. Assuming that higher-level instruction has only the structure implied by Figure 1--that is, higher-level open circles embrace no structure of their own--being simply integrative with respect to the lower-level skills that they subsume--then a Lesson 3 or Lesson 6 failure on an appropriate test for skills integration may be due either to the failure of Level 2 instruction or to Level 1 failure referencing to either of two subsumed skills. Lesson 14 failure, according to the same reasoning, may be due to five sources of failure. Let us assume that mainline Level 2 instruction will contain as many instructional segments as there are subsumed Level 1 skills. Thus, the Level 2 mainline sequences are:

Lesson 3: $IN_1\text{-}PS_{3a} + IN_1\text{-}PS_{3b} + EV_1\text{-}PS_3$

Lesson 6: $IN_1\text{-}PS_{6a} + IN_1\text{-}PS_{6b} + EV_1\text{-}PS_6$

Lesson 14: $IN_1\text{-}PS_{14a} + IN_1\text{-}PS_{14b} + IN_1\text{-}PS_{14c} + IN_1\text{-}PS_{14d} + EV_1\text{-}PS_{14}$

Particularly when a lesson has as many potential program skills sources of difficulty as Lesson 14 does, we either need to put EV-H_2 into the instructional system to aid pinpointing of the source or we need to increase length of the mainline test. For present purposes, we do the latter. Let $EV_1\text{-}PS_{14}$ give way to the two segments $EV_1\text{-}PS_{14a}$ + $EV_1\text{-}PS_{14b}$. Hence, the Level 2 mainline storage requirement is two 3-seg- program elements plus one 6-segment program element, or two 6-minute and one 12-minute program elements.

We can continue to live with the decision to employ only two alternate versions of Level 1 sequences if we are willing to assume that Level 1 sources implicated at Level 2 must arise simply due to "forgetting" or a need for refresher instruction. This convenient assumption makes it possible then to recycle to one of the earlier versions even though both may formerly have been used. However, if the source is found to be at Level 2, then an alternative version of the Level 2 sequence will be needed. This requirement is identical to that for Level 2 mainline instruction (see preceding paragraph).

Only Lesson 7 is Level 3. It integrates the subsumed skills PS_3 and PS_6. Treating it as we have Level 2, then mainline and looping instructional requirements each will be on the order of what we have shown at Level 2 for Lessons 3 and 6. Thus, the overall storage requirement is for two 3-segment program elements, or two 6-minute elements.

Only Lesson 9 is Level 4. Since it too integrates two skills--PS_7 and PS_8--it also imposes an overall storage requirement of two 3-segment program elements, or two 6-minute elements. Only Lesson 15 is Level 5. It also integrates two skills--PS_9 and PS_{14}--and so also imposes an overall storage requirement of two 3-segment program elements, or two 6-minute elements.

Like Level 1, Levels 2-5 each will require one alternative test per lesson that can be used independently of instruction following recycling to lower-level instruction. This adds five 1-segment (2-minute) and one 2-segment (4-minute) EV program elements to the storage inventory. Table 1 summarizes the overall storage requirement.

Table 1

Number of Program Elements, Element Lengths, and Normal-Play Minutes for Mainline and Supplemental Instruction

	Lessons	No. Program Elements	Element Length Segments	Total Normal-Play Minutes
Mainline				
EMS	0	4	1	8
Level 1	9	27	3	162
Levels 2-5	6	5	3	30
		1	6	12
Subtotals		(37)		(212)
Supplemental				
EMS	0	8	4	64
Level 1	9	27	3	162
		9	1	18
Levels 2-5	6	5	3	30
		5	1	10
		1	6	12
		1	2	4
Subtotal		(56)		(300)
Total	15	93		512

Since the illustration is an absolute bare-bones one, guaranteed to minimize storage-retrieval requirements when several Ss receive interactive instruction of moderate extent, we further assume that a single pace of instruction is employed, with variation in instructional duration occurring in consequence of differential supplementation of mainline instruction across Ss. Thus, each S will receive 14 minutes of mainline instruction per day (212/15) on the average. This instruction will feature presentation of 2.5 program elements per day (37/15) on the average. Allowing 30 seconds for switching from one element to the next will consume an additional minute on the average. Assume now that the S having the highest acquisition rate will manifest a mainline to formal looping ratio of 4:1 and that the S having the lowest acquisition rate will manifest a 1:1 ratio. In consequence, a child whose acquisition rate is average with respect to the illustrative program will manifest a 2.5:1 ratio if the distribution of rates is symmetrical. All Ss will average 15 minutes per day in mainline instruction. An S having average acquisition rate for the program will spend 6 additional minutes in formal looping instruction. Let us further assign to this S 4 minutes per day of informal instruction and evaluation consonant with his going beyond the limited formal instructional materials that the illustrative program makes available for supplemental purposes. Thus, we define the 30-minute session for an S of average rate as consisting of 15 minutes of mainline instruction, 6 minutes of formal supplemental instruction, 4 minutes of informal supplemental instruction, and 5 minutes of break time (positioned by E in the day's instructional sequence on the basis of his perception of S's needs). The consequence is that an S having an average rate will complete the program in 15 30-minute periods.

The highest-rate S will complete as much as the S of average rate in 21.25 minutes. If he also uses 5 minutes per day for breaks, then he will complete the program in 12.75 days. The lowest-rate child will complete as much as the average rate child in 40 minutes. If he also uses 5 minutes per day for breaks, then he will complete the program in 24 days. Accordingly, the 13th day will be a worst case day for program storage. The highest rate child will complete the program that day, while the slowest rate child will be only half way through at the beginning of the day.

The program is so devised that it is theoretically possible that any of the supplemental program elements might be required at any point in instruction--a condition I feel typically will prevail in interactive instruction. Hence, only mainline storage of program elements can be deleted on the worst case day. At the beginning of Day 13, the slowest rate child should have completed approximately 106 minutes of instruction. Completion of Lesson 7 signifies negotiation of 108 minutes of the mainline treatment. This removes 18 3-segment program elements from the Table 1 inventory. What remains as the worst day storage requirement is shown in Table 2.

Table 2

Worst Day Storage Requirement

	No. Program Elements	Element Length, Minutes	Total Normal-Play Minutes
	18	2	36
	1	4	4
	46	6	276
	8	8	64
	2	12	24
Total	75		404

In consequence of an extreme accommodation to the system, we have managed to bring the worst case audio program storage requirement down to a level that is consonant with the contractor's view of audio storage as 96 programs. All that we need to do with this simplified illustration to make it exceed 96 programs on a worst case day is adopt the view that supplemental materials should reflect two alternative versions to the mainline version, rather than one. Removing supplemental EMS elements, this adds 48 program elements. Added to the 75 for a worst day, the system becomes overburdened in light of the contractor's view of 96-program audio storage, for the worst day requirement now becomes 123 program elements.

Whatever may be true for audio is truer still for video. The 16-file view of video storage simply will not do.

The Looping Requirement

It remains to determine how the modest looping requirement inherent in the oversimplified illustration compares with the contractor's naive view that "first-level branching" epitomizes interactive instruction. The simplest (although probably not the most effective) approach to contingent instructional path specification for the illustrative program has H_0 evaluated first. $+H_0$ steps S along the mainline. $-H_0$ leads to evaluation of H_1. $+H_1$ leads S into supplemental EMS instruction with exit to a second (off-mainline) version of the instructional-evaluative program element for which $-H_0$ was obtained. $-H_1$ is interpreted as $+H_2$, leading S into the supplemental second version of the program element for which $-H_0$ was obtained. If system software will not support this degree of contingent advance, then that portion of the software that addresses the looping requirement simply will be useless.

Working Paper 7

PRESPECIFIED EVENT SEQUENCES IN INSTRUCTIONAL EXPERIMENTS: IMPLICATIONS FOR IDCMS (TN 1-72-07)

Joseph F. Follettie

Event sequences usually are prespecified in experiments. During execution of an experiment, E controls experimental events independently of characteristics of monitored performance. A minor exception to this rule is that S can cause experimental events to occur somewhat more quickly than the limiting slowest speed at which they are programmed to occur if he responds in less time than event programming allows. Instructional experiments at SWRL increasingly allow S to speed the pace of events by responding quickly. However, programmed response time tends to be a sufficiently minor component of most such experiments that the effect of differential response speed is to spread S's only modestly over an experimental event sequence when they start out at the same point in the sequence during a given session--e.g., 25-30 minutes. Even so, when two or more Ss begin from the same point in an event sequence and negotiate the event sequence at the same time, it is inevitable that they soon will be at different points in the sequence when rate of advance of the event sequence is made conditional on response speed. Hence, an event-control system applicable to execution of instructional experiments necessarily will allow the different Ss who participate in an experimental session under system control to proceed through the sequence at different rates. The SWRL Instructional Development Control and Monitoring System (IDCMS) has this capability; it allows six Ss participating in a given session to proceed at different rates through a given event sequence.

Just how far such a capability extends remains to be determined. For example, the difference between an event sequence that will allow six Ss of an instructional experiment each to have a minor effect on rate of advance of events while participating in the experiment for 25 minutes and a series of event sequences, one applicable to each of six Ss participating in such a session, is on the order of a fivefold or sixfold difference in the storage-retrieval-reproduction-control burden placed on the system. The optimal instructional experiment routinely would control for time-of-day effects. To do so requires representing each treatment equally often at each time of day in which sessions are scheduled. Thus, we must eventually require systems of the IDCMS type to serve at least as many Ss as there are treatments in the instructional experiment and to (partially) control event occurrences belonging to as many different sequences as there are treatments in the experiment. Thus, if the experiment features four treatments, then the requirement for input-output terminals is four or a multiple of four and the requirement for control span is four session-long event sequences. If the experiment features six treatments, then the requirement for terminals is six or a multiple of six and that for control span is six such

sequences. The system's capability for effecting (partial) event control should be evaluated for the illustrative six-treatment instructional experiment to be described.

Although response-monitoring functions will be vested in E in remarks that follow, IDCMS has a response-monitoring capability for certain types of responses not of interest here. The system's capability for monitoring responses contrasts with its capability for monitoring its own event-control and response-monitoring behavior. In the event-control domain, an assumption that the system can control an event sequence of interest referencing to several Ss served at the same time is an assumption that the system can monitor its own event-controlling behavior consonant with required event control. It is redundant to speak of an event-control capability and a capability for monitoring event-controlling behavior consonant with event-control.

Current instructional experiments typically are conducted in a manual mode; this term simply signifies that E is considerably burdened with response-monitoring and event-controlling activities during the conduct of an experiment. The consequence typically is a series of costly compromises which, on the one hand, reduce the usefulness of a datum and, on the other hand, make the cost of a datum so high that one seldom can afford the volume purchases of data that resolution of complex instructional problems requires. An optimal alternative to manual mode execution of instructional experiments is automatic mode execution. Study execution would be in automatic mode if E's presence in the response-monitoring and event-sequencing roles were unnecessary. Moreover, the well-documented advantages that machines enjoy over people when complex clerical behaviors are required should insure a diminution in costly compromises during study formulation as one moves away from manual mode execution and toward automatic mode execution. IDCMS will not make E's presence unnecessary, but it promises to relieve E of some event-controlling (and, in some situations, response-monitoring) functions. In consequence, the system promises to move instructional experimentation in the direction of automatic mode execution--an important implication of which is that such experimentation can increase in complexity as the phenomena under study warrant. The primary purpose of this paper is to specify a minimal set of characteristics that IDCMS must possess to relieve E of an appreciable amount of the event-control portion of his current manual-mode, study-execution burden. We do this by indirection. That is, an instructional study now in formulation is described well enough to guide those charged with system exploitation and evolution concerning how the system will be used in support of SWRL instructional research. The illustrative study will be executed in manual mode; we ask the system to support execution of such studies in such a way that study effectiveness will be enhanced through diminution of a need to make compromises stemming from event-controlling limitations of E and study efficiency will be (considerably) enhanced in the cost-return sense.

A study now being formulated by John Koehler illustrates the sort of instructional experiment we wish to be able to prepare for and execute routinely and cheaply at SWRL. Unlike experiments that typify the instructional research domain, the Koehler study rises to the challenge posed by the instructional problem, which is complex. Even so, manual-mode resources have necessitated some compromises with what we view as an effective response to the problem; manual-mode study execution precludes the rate of return in findings that we must have to reach definitively effective instructional designs in less than the long term. An acceptably useful IDCMS will permit both the study of more-complex event sequences than the Koehler study will evaluate in manual mode and the pursuit of a sharply accelerated rate of return in findings per unit resources.

Although the emphasis below will be on sequencing-control of experimental events, IDCMS has other functions that bear only slightly less on improved effectiveness-efficiency of instruction research. A few comments on these other functions are in order. All visual displays to be used in the Koehler study are alphanumeric (or, even more narrowly, "alphaic," since each is composed exclusively of at most four alphabetic characters). Research efficiency would be enhanced if the system's character generater could be used to produce camera-ready materials underlying the loading of such materials into the system's video (frame) storage. While the matter requires evaluation, my guess is that the system in Version 1 configuration will have such a capability.

A rough estimate is that the Koehler study will require at most 400 unique visual displays (slides in projector terminology, frames in system terminology). Manual mode execution of the study necessitates much duplication of the basic set of 400 frames. That is, to hold E's event-controlling burden within reasonable bounds, it is necessary that a set of approximately 1750 slides per E be prepared, sorted, and stored for use during study execution. While the study as formulated will process Ss four at a time, if it were executed in manual mode under conditions comparable to execution under partial system control, it would process Ss six at a time. To do this would necessitate that 10,500 slides be prepared, sorted, and stored for use during study execution. Actual manual-mode execution will necessitate that each basic slide on the average be duplicated on the order of 17 times; manual-mode execution that is comparable to execution under system support would necessitate that each basic slide on the average be duplicated on the order of 26 times. While the matter requires evaluation, my guess is that the system in Version 1 configuration would necessitate no duplication of visual displays whatsoever. That is, even if it were required that a given visual display be placed in the system's visual store on more than one track, the same single camera-ready display could be used for this purpose. However, duplication even in the sense of multiple storage addresses, if required, will be minimal, because each S is provided with

a video buffer to which the frame in master video storage can be duplicated. In consequence, if the system can produce Koehler study visual displays in camera-ready form, then if the cost of a single display is 50 cents, the study under partial system control would cost $200 for materials preparation, while a comparable study in manual mode would cost over $5000 for materials preparation. Unfortunately, this is not the entire cost of large materials requirements based on a high level of duplication. Inevitably, response time is a function of the materials requirement. In some sense, materials requirements of the magnitude inherent in the Koehler study necessitate waits that set back study execution. In some sense, this necessitates in turn that specialized professional staff will be less effectively employed than would otherwise be the case.

While the Koehler study tends to fall at the upper end of a scale for complexity of instructional research as this is conducted in contemporary research settings, it tends also to fall at the lower end of such a scale when the scale is defined on SWRL requirements for information that resolves germane instructional problems. Hence, the quantitative implications of the Koehler study for audio and video storage must be taken as underestimates of the system-burdening requirements that such a system must be able to handle in the years immediately ahead. A temptation was for the most part resisted to make the illustrative study more complex than the Koehler study because it is more difficult to argue away the inconvenient facets of actual work than it is to argue away the inconvenient facets of hypothetical work.

The Koehler study features six instructional treatments, each applied to one treatment group of 12 Ss over three successive test-train-test cycles. Ss are K-level. The Koehler schedule calls for preliminary entry skills evaluation on Day 1, Cycle 1 training on Days 2-3, Cycle 2 training on Days 5-6, Cycle 3 training on Days 8-9, and intervening post-training and entry skills testing on Days 4, 7, and 10. One of the few liberties we take with Koehler's formulation is to collapse the schedule to eight days, with testing occurring on Day 1 and toward the end of Days 3, 6, and 8. The intent here is not pedagogical, but rather to stress the system a bit more than the Koehler schedule would.

It is assumed here that the system should store, consonant with momentary retrieval and reproduction to S's audio and video buffers, all of the materials that will be used during one experimental day. This assumption frees us to schedule sessions tightly so as to fully exploit the school day; this, in turn, creates most-favorable system amortization conditions. The object is to schedule the Koehler experiment so that the worst-day storage requirement can be identified. The basis for such scheduling, while straightforward, involves production of much detail. This detail, in the form of flowcharts and tables, is presented in Appendix A, together with a more extensive description of the study than will be provided here. Koehler has instituted a two-way control for amount.

of training across treatments within cycles: a) number of training responses per treatment per cycle and b) associated training time. We use his values for number of training responses and a framework that appears compatible with IDCMS to estimate associated training times. While training times as derived in Appendix A are characterized as those for an S whose response speed is a median value, the possibility of variation in response speed in a study of the Koehler type is over a constrained range. Hence, these times do not probably seriously underestimate training times for the slowest-responding study S. The training-testing schedule to be presented assumes the following:

1. Thirty-minute sessions wherein six minutes are given over to housekeeping events and rest breaks. Hence, study event time is on the order of 24 minutes.

2. Twelve consecutive sessions per school day, beginning at 8:30 a.m. and ending at 2:30 p.m., with six Ss participating in each session. Hence, the study N of 72 Ss in toto will participate during each experimental day. (This is no mere extraneous requirement. A persistent problem of most instructional research is that different batches of children enter the experimental situation at different points in the school year. This is relatively unimportant at higher-grade levels but can be devastating at the K level. While confounding procedures exist whereby one can control for effects of different amounts of prior school experience, this can only be done at a cost in error variance; this cost can also be characterized as a cost in study efficiency.)

Since Day 1 performance conditions assignment to groups, Day 1 experimental events will be confined to a preliminary entry skills test. In consequence, 10-minute sessions will characterize Day 1. Sessions on all other days will be 30 minutes. Table 1 shows how the Koehler study would be scheduled consonant with the data presented or derived in Appendix A and foregoing assumptions. The derived training times reflected in Table 1 meet Koehler's requirement that each treatment group receive about the same amount of training during a given cycle (see also Tables A-21 and A-22); they also appear consonant with Koehler's views on training time expenditures that the study will encounter.

For many compelling reasons that go beyond the purview of this paper, the system must eventually evolve to a point wherein it permits on-line rapid composition of programs from the fewest number of elements that are consonant with such an objective. However, Appendix A defines the program as a unitary entry in storage such that the requirments on the system are only those of retrieving a program on command and reproducing it to the buffer of the S who is ready to negotiate the program. Events within a given program occur in fixed sequence. Table 1 reveals that, for a given treatment group, the programs that apply to the group themselves will be negotiated in fixed sequence. All that will vary across two or more Ss negotiating a given stretch of a treatment set of programs during a given session is that these Ss may vary just a

Table 1

Training-Testing Schedule, by Treatment Group and Type of Program[a]

Day	Type of Program	Treatment Group					
		S1	B1	C1	S2	B2	C2
1	ET-1	7.5	7.5	7.5	7.5	7.5	7.5
2	1/T1.0	3.3		3.3	3.3		3.3
	1/T2.1	19.0		9.7			
	1/T2.2a				9.6		5.0
	1/T2.2b				9.7		5.1
	1/T3.0a	2.1	2.1	2.1	2.1	2.1	2.1
	1/T3.0b		2.1	2.1		2.1	2.1
	1/T3.0c		6.8	6.8		6.8	6.8
	1/T4.1		13.0				
	1/T4.2a					12.5	
	Totals	24.4	24.0	24.0	24.7	23.5	24.4
3	1/T3.0b	2.1			2.1		
	1/T3.0c	6.8			6.8		
	1/T4.1		9.1	9.7			
	1/T4.2a						6.5
	1/T4.2b					9.7	3.5
	PT-1	6.7	6.7	6.7	6.7	6.7	6.7
	ET-2	7.5	7.5	7.5	7.5	7.5	7.5
	Totals	23.1	23.3	23.9	23.1	23.9	24.2
4	2/T1.0	5.8		5.8	5.8		5.8
	2/T2.1	18.2		18.5			
	2/T2.2a				17.5		8.2
	2/T2.2b						5.0
	2/T2.2c						5.1
	1/T3.0a		2.1			2.1	
	1/T3.0b		2.1			2.1	
	1/T3.0c		6.8			6.8	
	2/T4.1		13.0				
	2/T4.2a					12.5	
	Totals	24.0	24.0	24.3	23.3	23.5	24.1

[a]All entries are in minutes.

Table 1 - continued

Day	Type of Program	Treatment Group S1	B1	C1	S2	B2	C2
5	2/T2.1	18.4					
	2/T2.2b				11.8		
	2/T2.2c				9.7		
	1/T3.0a	2.1		2.1	2.1		2.1
	1/T3.0b	2.1		2.1	2.1		2.1
	1/T3.0c	1.4		6.8			6.8
	2/T4.1		24.0	13.0			
	2/T4.2a						6.5
	2/T4.2b					11.8	5.0
	2/T4.2c					12.2	1.5
	Totals	24.0	24.0	24.0	25.7	24.0	24.0
6	1/T3.0c	5.4			6.8		
	2/T4.1		5.7	5.5			
	2/T4.2c					6.3	8.0
	PT-2	6.7	6.7	6.7	6.7	6.7	6.7
	ET-3	7.5	7.5	7.5	7.5	7.5	7.5
	3/T1.0	4.6		4.6	3.6		2.5
	2/T3.0a		2.6			2.6	
	Totals	24.2	22.5	24.3	24.6	23.1	24.7
7	3/T1.0				1.0		2.1
	3/T2.1	18.7		9.1			
	3/T2.2a				11.3		5.1
	3/T2.2b				5.2		2.8
	3/T2.2c				3.7		2.1
	2/T3.0a	2.6		2.6	2.6		2.6
	2/T3.0b	2.6	2.6	2.6		2.6	2.6
	2/T3.0c		8.8	8.8		8.8	6.7
	3/T4.1		12.6				
	3/T4.2a					12.5	
	Totals	23.9	24.0	23.1	23.8	23.9	24.0

Table 1 - continued

Day	Type of Program	Treatment Group S1	B1	C1	S2	B2	C2
8	2/T3.0b				2.6		
	2/T3.0c	8.8			8.8		2.1
	3/T4.1		11.1	9.7			
	3/T4.2a						6.5
	3/T4.2b					11.3	5.1
	PT-3	6.7	6.7	6.7	6.7	6.7	6.7
	Totals	15.5	17.8	16.4	18.1	18.0	20.4

little in time to completion of that stretch. Table 2 shows day-to-day storage requirements that the Koehler study, as scheduled in Table 1, requires. We take these requirements as those that must be met on a momentary basis; that is, the system must be able to retrieve and duplicate to an appropriate buffer any of the materials applicable to an experimental day at any time during any session of that day.

The worst case days for number of audio programs are Days 4 and 7. In either case, the system must store in audio master reproduction 40 short programs. It is my impression; but requires evaluation, that such a storage requirement underlying quick retrieval and duplication to an appropriate audio buffer is well within system capability in current configuration. A matter that should be explored more fully is whether the system can handle the program repetitive and interrupt functions described in Appendix A. These programs are designed so as to require audio buffer to stop between subprograms and to rewind to program beginning as multiple trials of the Koehler study require.

The worst-case day for amount of audio tape--in normal-play minutes--is Day 6. It is apparent that all required audio programs are at least fivefold shorter than the system will allow. Hence, the information on program length is less crucial than that on number of programs. However, it is instructive in the sense that it documents the research staff contention that the present hardware configuration of IDCMS wastes most of the audio program storage that it provides, when judged against SWRL research requirements. We may find it necessary in time to require the system to store more than 96 different audio programs. It is doubtful that we ever will require it to store more than 96 x 15 (or 25) minutes worth of audio programs. The 15 (or 25) minute program length engineered into the system in present configuration is virtually useless when instructional research is to be supported.

Day 6 is a worst-case day for video frame storage. While the 106 frames requiring storage on Day 6 poses no problem in light of the 1800-track video storage capability, the 16-file view of video storage that underlies current system engineering is inconsonant with the Table 2 Day 6 requirement that these 106 frames reference to 38 different audio programs. While many of these programs share the same video frames (the 12 PT-2 programs, for example), these frames must appear in different orders in the different programs. A central matter requiring early evaluation is whether the system in current configuration can yield the video correlation to audio programs which Day 6 (and, in fact, every other experimental day of the Koehler study) requires. If not, then priority must be given to securing such a capability, for it becomes a bottleneck that sharply constrains the uses to which the system can be put.

While not of central concern here, it requires comment that a system that is consonant with requirements summarized in Tables 1 and 2 surely would effect a variety of economies for which instructional research would be a prime beneficiary. Remarkable staff savings in the

Table 2

Per Day Materials Storage-Retrieval Requirement[a]

Day	Type of Program	Number of Programs	Program Length (min)	V Frames /Program	Total Tape (min)	Total V Frames
1	ET-1	6	3.1	28	18.6	28
2	1/T1.0	4	.5	2	2.0	4
	1/T2.1	4	1.0	4	4.0	8
	1/T2.2a	4	1.5	6	6.0	12
	1/T2.2b	4	1.0	4	4.0	8
	1/T3.0a	4	.8	[6]	3.2	[12]
	1/T3.0b	4	.8		3.2	
	1/T3.0c	4	1.6		6.4	
	1/T4.1	4	1.0	4	4.0	8
	1/T4.2a	4	2.0	8	8.0	16
	Totals	36			40.8	68
3	1/T3.0b	4	.8	[6]	3.2	[12]
	1/T3.0c	4	1.6		6.4	
	1/T4.1	4	1.0	4	4.0	8
	1/T4.2a	4	2.0	8	8.0	16
	1/T4.2b	4	1.0	4	4.0	8
	PT-1	12	2.0	20	24.0	20
	ET-2	6	3.1	28	18.6	28
	Totals	38			68.2	92
4	2/T1.0	4	.9	4	3.6	8
	2/T2.1	4	2.0	4	8.0	8
	2/T2.2a	4	1.0	4	4.0	8
	2/T2.2b	4	1.5	6	6.0	12
	2/T2.2c	4	1.0	4	4.0	8
	1/T3.0a	4	.8	[6]	3.2	[12]
	1/T3.0b	4	.8		3.2	
	1/T3.0c	4	1.6		6.4	
	2/T4.1	4	2.0	4	8.0	8
	2/T4.2a	4	2.0	8	8.0	16
	Totals	40			54.4	80

[a]Bracketed frame entries are common to sets of T3.0 programs.

Table 2 - continued

Day	Type of Program	Number of Programs	Program Length (min)	V Frames /Program	Total Tape (min)	Total V Frames
5	2/T2.1	4	2.0	4	8.0	8
	2/T2.2b	4	1.5	6	6.0	12
	2/T2.2c	4	2.0	8	8.0	16
	1/T3.0a	4	.8	[6]	3.2	[12]
	1/T3.0b	4	.8		3.2	
	1/T3.0c	4	1.6		6.4	
	2/T4.1	4	2.0	4	8.0	8
	2/T4.2a	4	2.0	8	8.0	16
	2/T4.2b	2	1.5	6	3.0	12
	2/T4.2c	4	2.0	8	8.0	16
	Totals	38			61.8	100
6	1/T3.0c	4	1.6	6	6.4	12
	2/T4.1	4	2.0	4	8.0	8
	2/T4.2c	4	2.0	8	8.0	16
	PT-2	12	2.0	20	24.0	20
	ET-3	6	3.1	28	18.6	28
	3/T1.0	4	.7	3	2.8	6
	2/T3.0a	4	1.1	8	4.4	16
	Totals	38			72.2	106
7	3/T1.0	4	.7	3	2.8	6
	3/T2.1	4	1.5	6	6.0	12
	3/T2.2a	4	1.0	4	4.0	8
	3/T2.2b	4	.8	3	3.2	6
	3/T2.2c	4	.5	2	2.0	4
	2/T3.0a	4	1.1	[8]	4.4	[16]
	2/T3.0b	4	1.1		4.4	
	2/T3.0c	4	2.1		8.4	
	3/T4.1	4	1.0	4	4.0	8
	3/T4.2a	4	2.0	8	8.0	16
	Totals	40			47.2	76

Table 2 - continued

Day	Type of Program	Number of Programs	Program Length (min)	V Frames /Program	Total Tape (min)	Total V Frames
8	2/T3.0b	4	1.1	[8]	4.4	[16]
	2/T3.0c	4	2.1		8.4	
	3/T4.1	4	1.0	4	4.0	8
	3/T4.2a	4	2.0	8	8.0	16
	3/T4.2b	4	1.0	4	4.0	8
	PT-3	12	2.0	20	24.0	20
	Totals	32			52.8	68

form of more apt use of specialized personnel should result; these savings would be transformed into the higher rate of acquisition of germane findings we must have to achieve definitively effective instructional designs in less than the long term.

APPENDIX A

OUTLINE OF THE KOEHLER STUDY

A study now being formulated by John Koehler illustrates contemporary instructional experiments. While this study will be conducted in the manual mode during Spring 1972, we view it from a system control standpoint. We ask what characteristics a control-monitoring system must have to permit shifting most of the event control burden of manual mode execution of the Koehler study from E to the system.

The study compares alternative instructional treatments of a phonics approach to reading for effectiveness. Excepting that treatment groups are formed on the basis of comparable germane entry skills, no facet of the study is conditional on any characteristic of S's performance. Each of several types of response to be studied must occur in less than a generous amount of time. To minimize subsequent data reduction, E will (quickly) evaluate responses as these occur. Were the study under system control, E would feed the proper evaluative code to the system for recording immediately following each response.

The study occurs in three cycles. These cycles differ primarily in the lexical-phonemic structure of the words whose processing is to be instructed and tested. Cycle 1 features CVC words (e.g., SAP); Cycle 2, CCVC words (e.g., SNIT); Cycle 3, CVCC words (e.g., STNK). Each cycle is characterized by its own 3-segment entry skills test and its own 3-segment post-training test; the intervening training for a cycle consists of one of six training treatments each of which is further differentiated according to which of two sets of instructional materials is used.

ET and PT Tests

Denoting entry tests ET, post-training tests PT, test segments A, B, C, and cycles 1, 2, 3, then test notation is as shown in Table A-1.

All Ss receive all tests in the progression ABC. Segment C post-training subtests--PTC-1, PTC-2, PTC-3--are in two alternative versions, one appropriate to half of the Ss and the other appropriate to the other half. Otherwise, testing is identical across Ss, regardless of training. Described in order below are: a) ETA and PTA subtests and associated control requirements, b) other subtests, and c) cycle-by-cycle ET and PT storage requirements and the time it will take to test the 72 Ss used in the study when six Ss at a time are tested under system control. Testing and training time values are of interest because single day (or session) storage requirements stem from timing a study over sessions and days using such values.

Table A-1

Test and Subtest Notation and Progression of Activities

Cycle 1			Cycle 2			Cycle 3		
1	2	3	4	5	6	7	8	9
ET-1	Tng	PT-1	ET-2	Tng	PT-2	ET-3	Tng	PT-3
ETA-1		PTA-1	ETA-2		PTA-2	ETA-3		PTA-3
ETB-1		PTB-1	ETB-2		PTB-2	ETB-3		PTB-3
ETC-1		PTC-1	ETC-2		PTC-2	ETC-3		PTC-3

ETA and PTA Subtests

ETA subtests differ in form from PTA subtests only in that ETAs consist of an example followed by six test items, whereas PTAs dispense with the example. (Different items are used on the two subtests; however, this is a difference in content rather than form.) For purposes of analysis, the only significance of an example is that the subtest will consist of seven or six items.

Examples and test items all have the following form:

1. A video frame (V i.1) containing two printed words (W_1 on the left, W_2 on the right) is presented with an audio accompanyment (A i) that pronounces first W_1 and then W_2.

2. Then V i.1 is superseded by V i.2, wherein W_1 is at top left, W_2 is at top right, and W_3 is boxed at bottom center.[1]

3. S is required to attempt pronunciation of W_3--S_{Ri} = Try /W_3/. (Slant brackets indicate audio presentation or aural responding.) S is encouraged to make S_{Ri} as quickly as he can (communicated beforehand during an interval that precedes system control of the testing sequence). S is allowed at most 12 seconds from onset of V i.2 to complete the response.

4. As soon as S responds, E quickly evaluates the response and then immediately addresses the system. If the item is an example rather than a test item, then E presses an Ex button, signifying an order to advance without recording--E_{Ri} = Ex (a command). If the response is to a test item, E presses one of eight evaluation code buttons, signifying an order to advance, coupled with orders to record the code and a suitable response time measure--E_{Ri} = Code i.[2]

There are three versions of each ETA and PTA, one for each cycle of the study. Desirably, the order of test items would be varied for

---

[1]The Koehler study will make do with a single visual frame--V i rather than V i.1 + V i.2--because this lessens the materials preparation and frame presentation burdens. IDCMS should be required to make such a compromise unnecessary.

[2]The Koehler study will not collect response time measures because to do so would add to an already extensive burden placed on E. IDCMS should be required to make such relinquishment of relevant data unnecessary.

each version of each subtest.[3] Treatment group size is consonant with our providing six orders for each six-item subtest--e.g., 1st: 123456, 2nd: 564312, 3rd: 216543, 4th: 345621, 5th: 652134, 6th: 431562.

Required that each of 3 ETAs and 3 PTAs (and all other subtests) occur in 6 versions that differ only for test item order, one's choices are to require the system to compose tests on-line (or nearly so) from item files for each subtest or to store the different versions as fixed sequences. The system in Version 1 configuration cannot do the former unless somewhat modified. Whether it can do the latter without modification depends on how many audio programs and associated video frames are to be stored, together with extent of these files. The paper to which this analysis is appended summarizes storage requirements--e.g., the requirement for one day (or session) of study execution. Below we will evaluate the normal-play extent of audio tape and the number of video frames that the ETA and PTA requirements outlined above entail.

Figure A-1 shows the event control scheme over time for an ETA or PTA item. According to the scheme, each item's audio component will require 9 seconds of single-track tape (with message and 55 Hz codes merged), or 33.75 inches of tape. Associated with this audio element will be 2 video frames. Thus, a given version of ETA will use 63 seconds of tape and 14 video frames. Assuming file identification needs that are additional to the foregoing, then a version of ETA might be taken to require 1.2 minutes of audio tape and 15 video frames. We call such a tape an audio program. It is assumed that every presentation or presentation sequence will have an audio program that enters into system control of the presentation. This will be true whether presented stimuli are audio, video, or audio + video. A version of PTA will require 1.0 minutes of audio tape and 13 video frames. Based on 6 item orders multipled by 3 cycles, program requirements in support of Segment 1 testing are:

1. 18 audio programs of 1.2 minutes duration (21.6 minutes) + 45 video frames (15 frames x 3 cycles) for ETA.

2. 18 audio programs of 1.0 minutes duration (18 minutes) + 39 video frames (13 frames x 3 cycles) for PTA.

---

[3]The Koehler study will not vary test item order for subtests because to do so under manual administration would risk attributing a wrong order to a given S; E has too many other things to do during manual administration. IDCMS should be required to make such a compromise unnecessary.

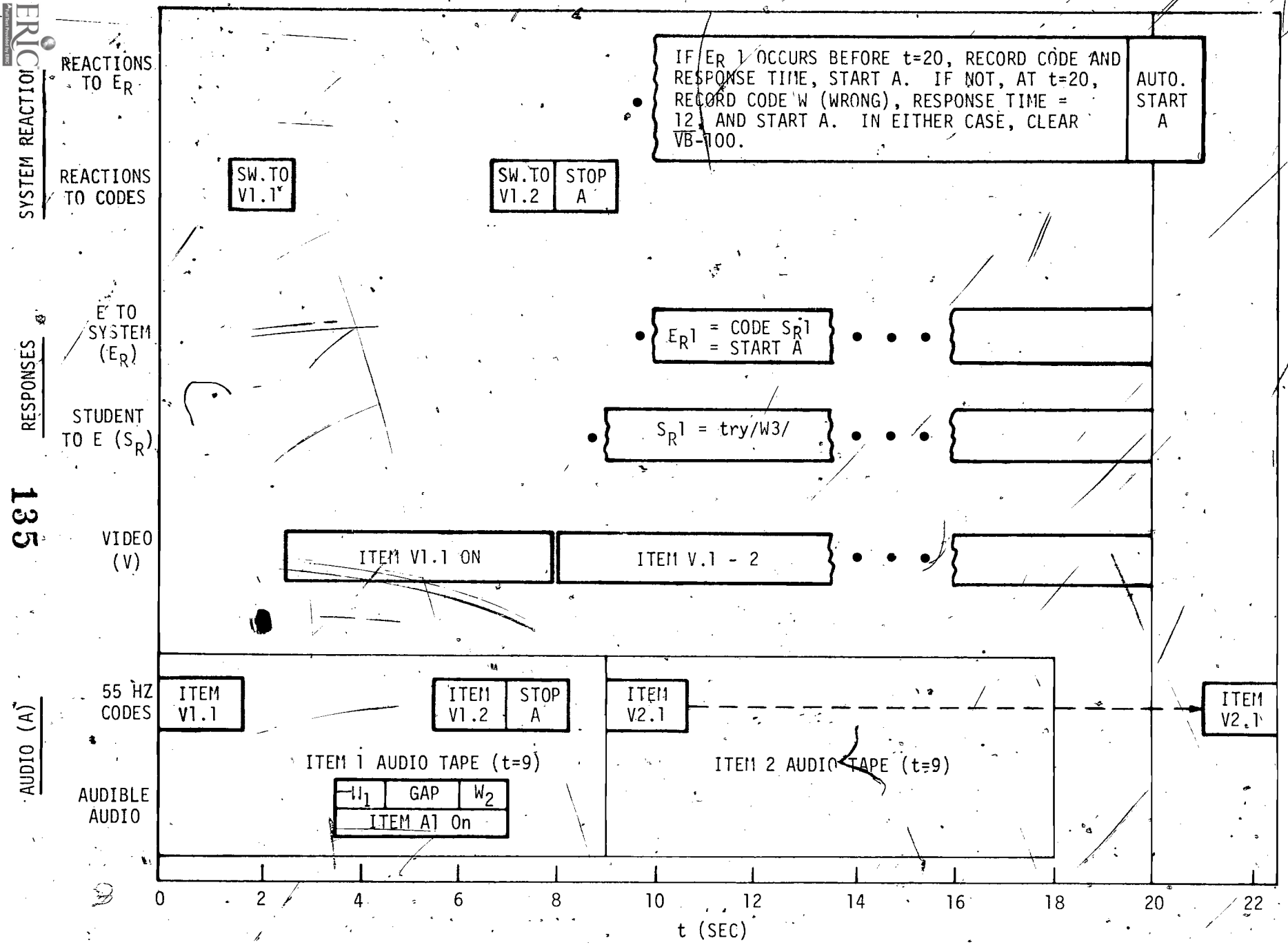


FIGURE A-1. SCHEMATIC REPRESENTATION OF EVENT CONTROL FOR ETA AND PTA ITEMS.

Other Subtests

All ETB, ETC, PTB, and PTC subtests will consist of 6 test items without a preceding example. Figures A-2 and A-3 show event control schemes for ETB and ETC items, respectively. Figures A-4 and A-5 show such schemes for PTB and PTC items. PTB items require different types of responses, depending on whether training features a single-letter or bi-part strategy (the training strategic factor of a 3 x 2 design for training treatments). PTC items require different types of item presentation, depending on whether training features a single-letter or bi-part strategy. PTC subtests therefore reflect two alternative versions per cycle, one that tests for effects of the single-letter training strategy and one that tests for effects of the bi-part training strategy.

Figure A-2 indicates that each ETB item will use 7.5 seconds of audio tape and 2 video frames. Hence, an ETB subtest will use 45 seconds of tape and 12 video frames. Using the reasoning applied earlier to ETA, we increase the requirement to 53 seconds of tape and 13 video frames. Figure A-3 indicates that each ETC item will use 8 seconds of audio tape--there is no video requirement. Hence, the ETC subtest will use 48 seconds of tape, which we increase to 56 seconds. Figure A-4 indicates that each PTB item will use 3.5 seconds of tape and 1 video frame. Hence, a PTB subtest will use 21 seconds of tape and 6 video frames, which we increase to 25 seconds of tape and 7 video frames. Figure A-5 indicates that each PTC item will use 3.5 seconds of tape--there is no video requirement. Hence, the PTC subtest will use 21 seconds of tape, which we increase to 25 seconds.

Program requirements in support of Segments 2 and 3 testing are:

1. 18 audio programs of .9 minutes duration (16.2 minutes) + 39 video frames for ETB.

2. 18 audio programs of 1 minute duration (18 minutes) for ETC.

3. 18 audio programs of .5 minutes duration (9 minutes) + 21 video frames for PTB.

4. 36 audio programs (3 cycle versions x 2 training strategy versions x 6 test item orders) of .5 minutes duration (18 minutes) for PTC.

Cycle-by-Cycle Summary

Table A-2 shows required program materials by cycle and test. It is worthy of note that the three segments of a test will occur in fixed sequence. Hence, the segments of ET tests can be stored as single programs (if stop codes are used at the ends of segments) for unitary retrieval and reproduction to audio buffer, thus reducing these programs

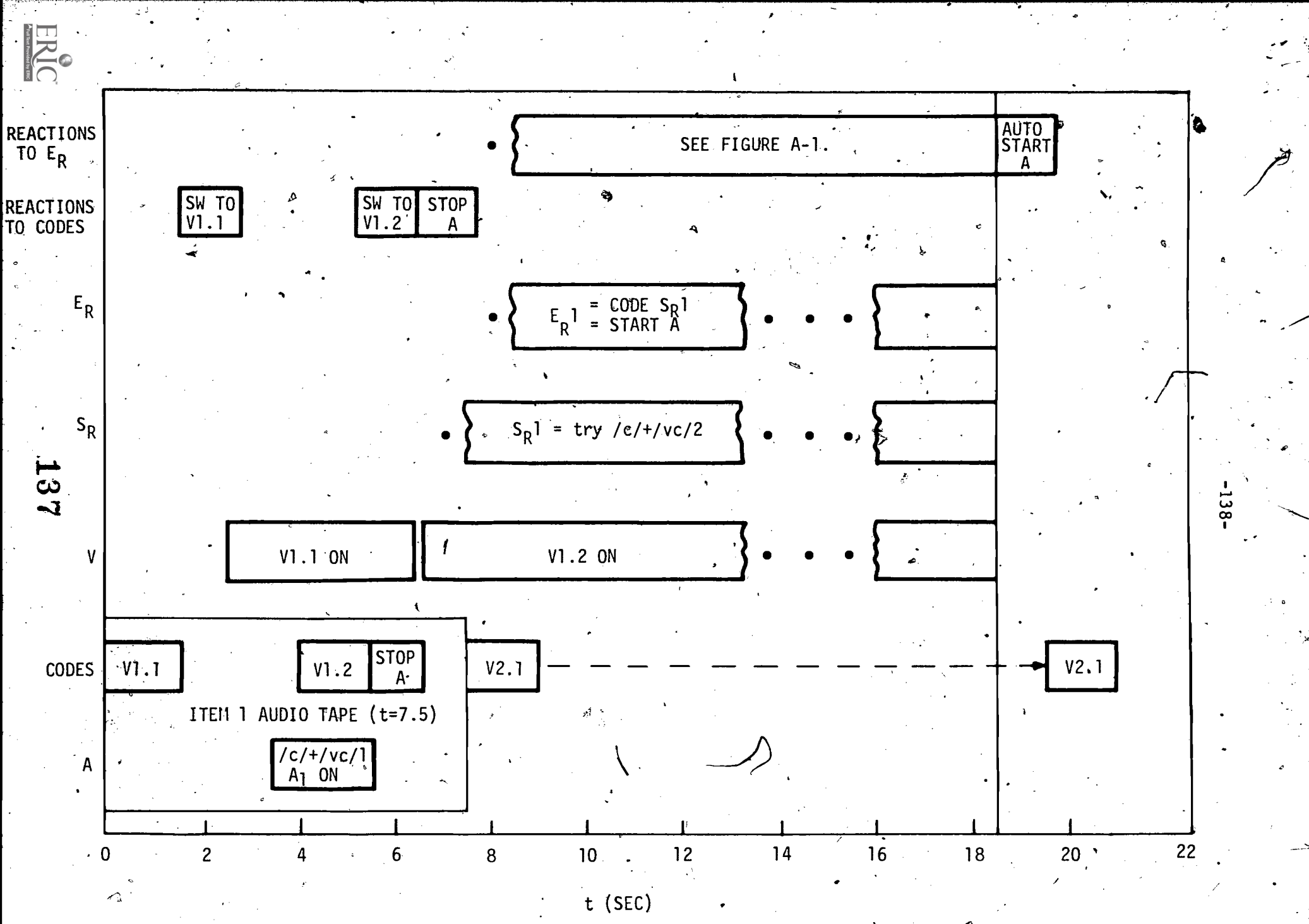


FIGURE A-2. SCHEMATIC REPRESENTATION OF EVENT CONTROL FOR ETB ITEMS.

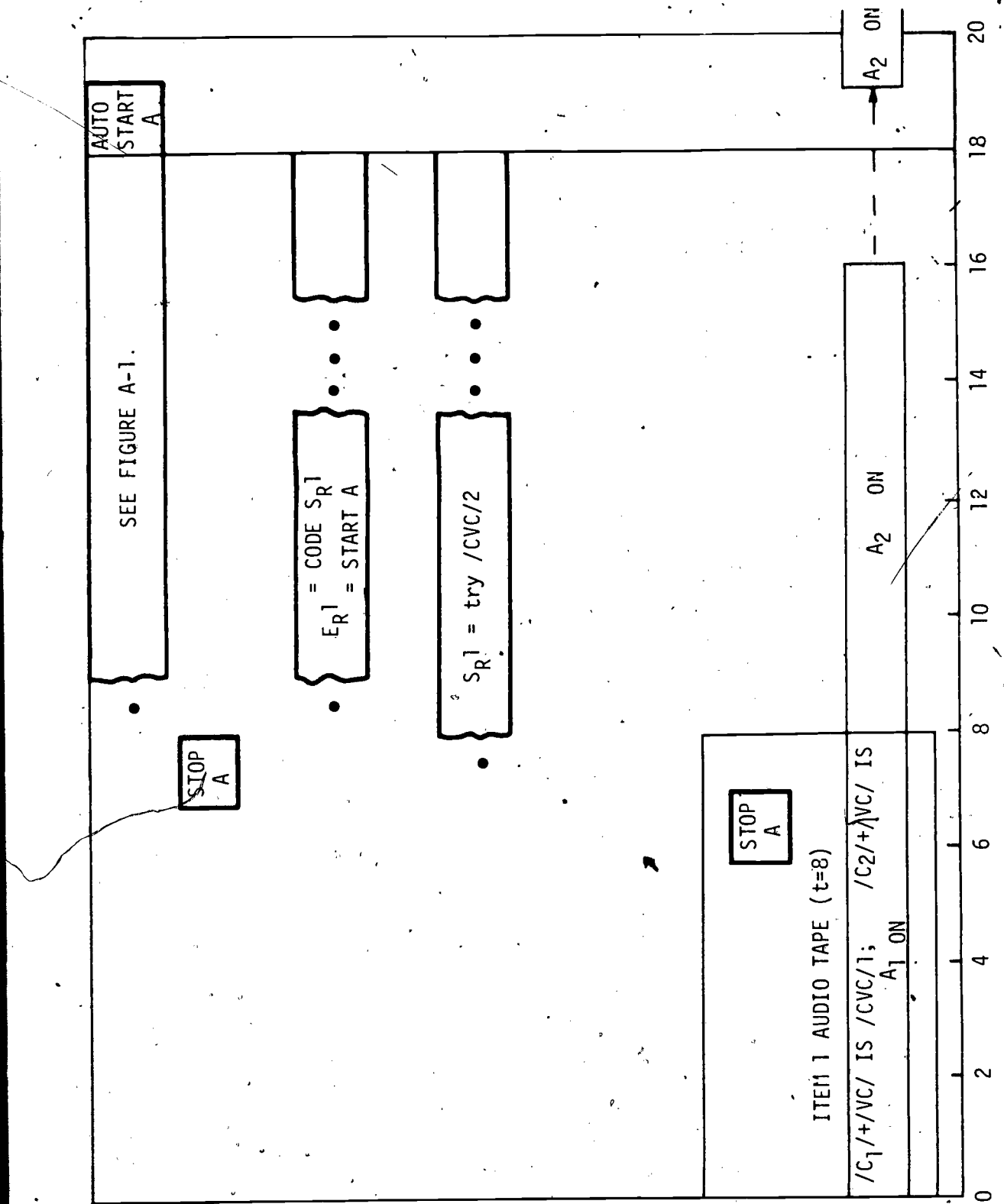


FIGURE A-3. SCHEMATIC REPRESENTATION OF EVENT CONTROL FOR ETC. ITEMS.

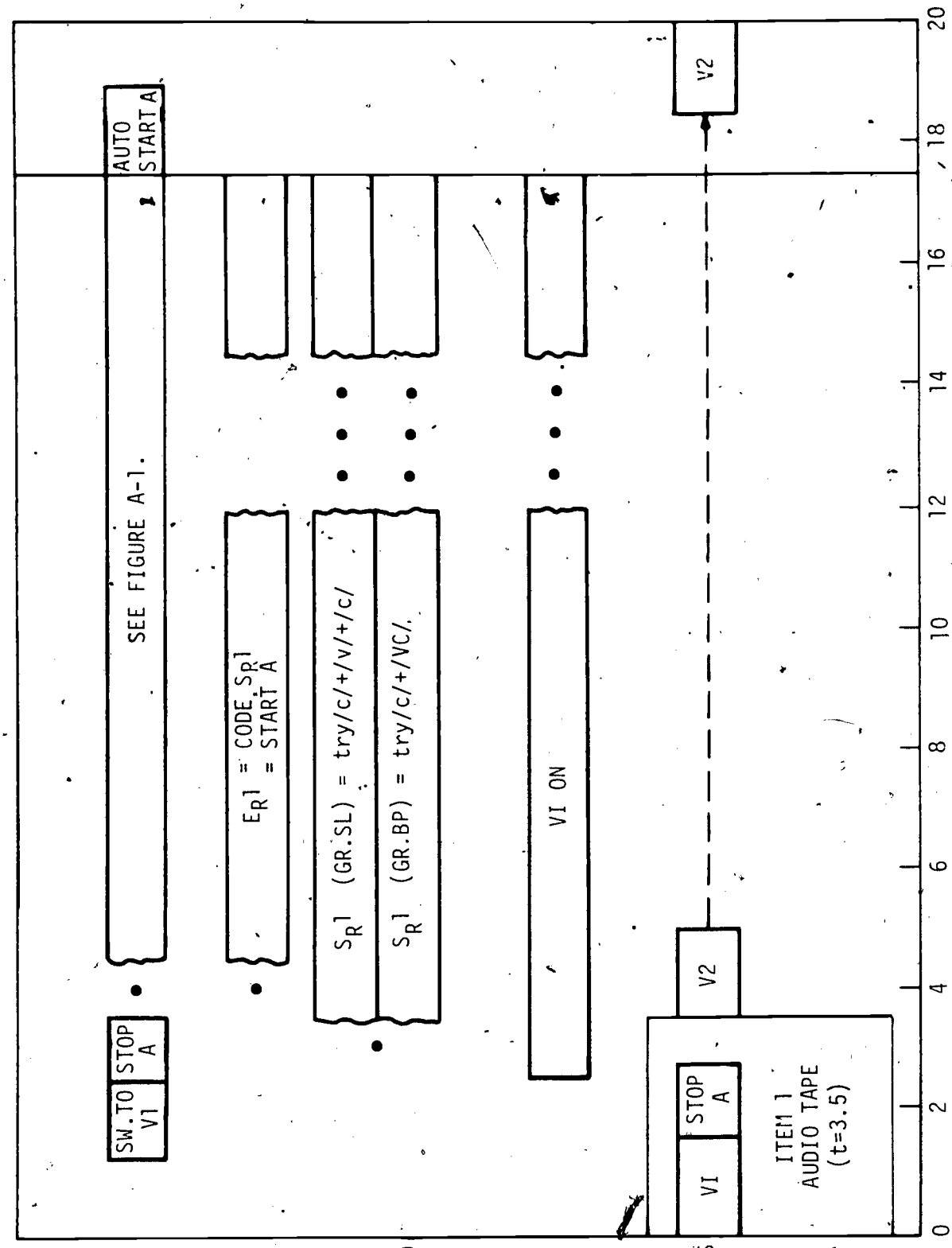


FIGURE A-4. SCHEMATIC REPRESENTATION OF EVENT CONTROL FOR PTB ITEMS.

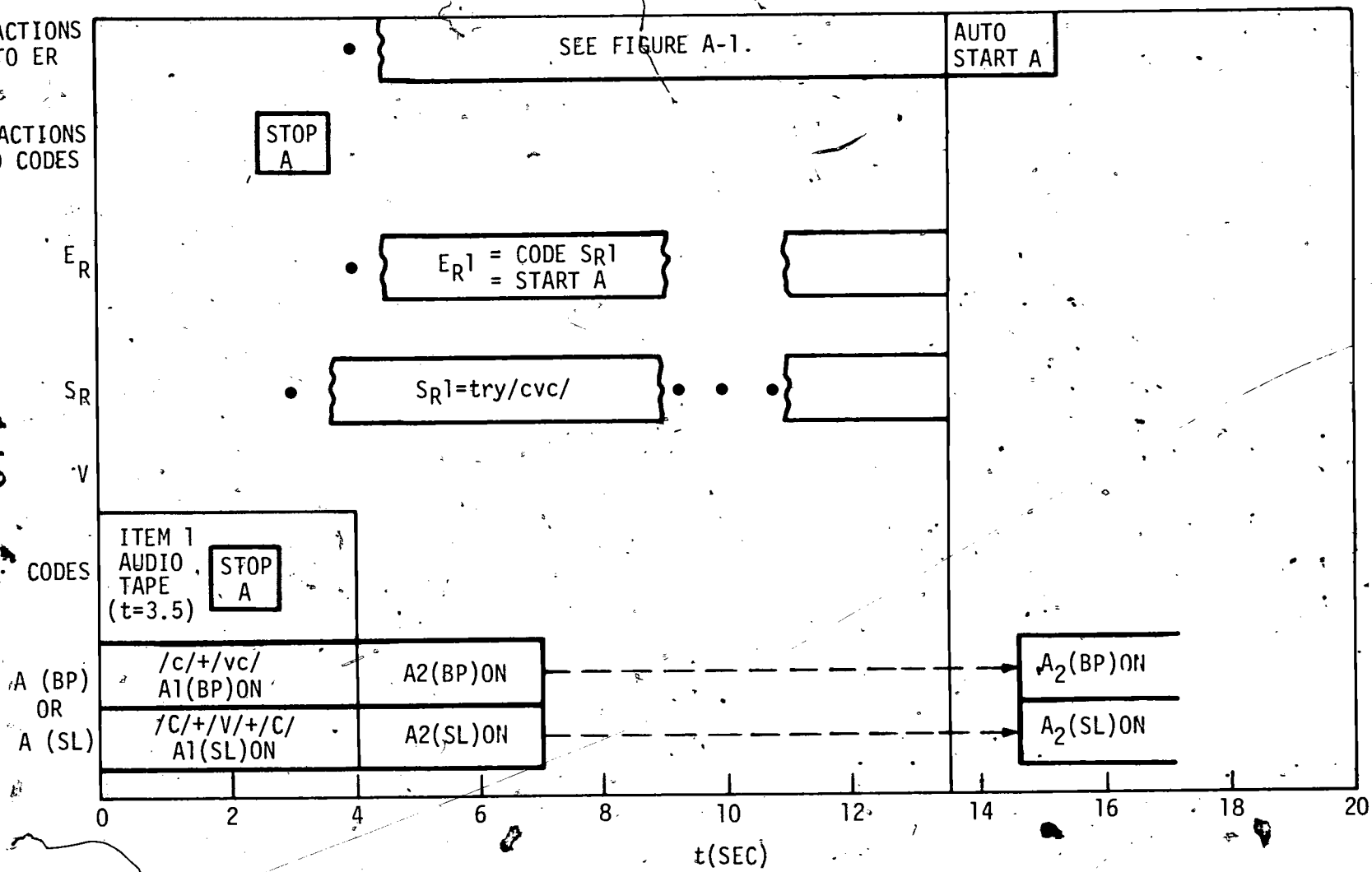


FIGURE A-5. SCHEMATIC REPRESENTATION OF EVENT CONTROL FOR PTC ITEMS.

Table A-2

ET and PT Program Materials, by Cycle and Type of Test

	Cycle 1		Cycle 2		Cycle 3	
	ET-1	PT-1	ET-2	PT-2	ET-3	PT-3
FIRST SEGMENT						
Programs x Minutes	6 (1.2)	6 (1.0)	6 (1.2)	6 (1.0)	6 (1.2)	6 (1.0)
Total Tape Minutes	7.2	6.0	7.2	6.0	7.2	6.0
No. Video Frames	15	13	15	13	15	13
SECOND SEGMENT						
Programs x Minutes	6 (0.9)	6 (0.5)	6 (0.9)	6 (0.5)	6 (0.9)	6 (0.5)
Total Tape Minutes	5.4	3.0	5.4	3.0	5.4	3.0
No. Video Frames	13	7	13	7	13	7
THIRD SEGMENT						
Programs x Minutes	6 (1.0)	12 (0.5)	6 (1.0)	12 (0.5)	6 (1.0)	12 (0.5)
Total Tape Minutes	6.0	6.0	6.0	6.0	6.0	6.0
No. Video Frames	0	0	0	0	0	0
TOTALS FOR TESTS[a]						
Programs	18	24	18	24	18	24
Total Tape Minutes	18.6	15.0	18.6	15.0	18.6	15.0
No. Video Frames	28	20	28	20	28	20

[a]If programs are defined on three-segment tests, rather than on single segments--which they can be because the order ABC is invariant--then ET tests yield 6 programs per cycle; tape length for these programs is 3.1 minutes per program. Consonant with the two different versions of PTC segements, PT tests yield 12 programs per cycle; tape length for these programs is 2 minutes per program.

to 6 in number (one per test item order), each consisting of 6.1 minutes of tape, referenced to a video file containing 28 frames whose sequencing will be a function of test item order. The same is true for PT tests, except that alternate training strategy versions of PTCs necessitate that there be 12 three-segment programs, each 5.0 minutes in tape length and all referencing to a video file containing 20 frames whose sequencing will be a function of test item order.

Average item negotiation times for ETA, ETB, ETC, and PTA subtests should be on the order of 15 seconds. Average item negotiation times for PTB and PTC subtests, respectively, should be on the order of 12 and 10 seconds. Exclusive of retrieval-reproduction intervals and administrative time not under system control that is used to give general instructions and effect housekeeping arrangements, an S on the average would use 15 x 18 seconds, or 4.5 minutes, to negotiate items of any ET test and 15 x 6 + 12 x 6 + 10 x 6 seconds, or 3.7 minutes, to negotiate items of any PT test. If we allow 3 minutes per test for administrative matters and concurrent retrieval-reproduction of programs to audio buffer, then each entry skills test on the average will use 7.5 minutes of S's (and so of the system's) time; each post-training test on the average will use 6.7 minutes of S's time.

Testing 72 Ss on a given entry test on the average will cost 12 x 7.5 minutes, or 1.5 hours, when tests are system-controlled and 6 Ss are tested at a time. Testing 72 Ss on a given post-training test on the average will cost 12 x 6.7 minutes, or 1.4 hours. The total testing program under these conditions will cost 3 x 1.5 hours + 3 x 1.4 hours, or 8.7 hours.

Assignment to Treatment Groups

Ss will be assigned to treatment groups on the basis of performance on ET-1. Each item will be scored for correctness of initial (i), medial (m), and terminal (t) features or portions of the response. A response will be scored R (correct) in the Koehler study if all portions of the response are correct and W (incorrect) if none are correct. It will be scored R_1-R_3 if two portions--im, it, mt--are correct and R_4-R_6 if one portion--i, m, t--is correct.[4]

[4]A truly automatic-mode system would both monitor and evaluate responses and would store both the monitored response and its evaluation code. A monitored response is one that is apprehended in the form given--the so-called raw datum. If S responds "No" to an item, then a monitor--whether human or mechanical--is said to have monitored that response if, on demand, the monitor can convey that S's response was "No." Latency or response time values also can be monitored. An evaluated response is one that is compared with a set of criterion specifications bearing on response accuracy, speed, or both. The simplest evaluated response

While Koehler presently can specify the sorts of "skills profiles" that will occasion controlled assignment of Ss to treatment groups, he will not know until Cycle 1 entry skills testing is completed just what sorts of skills profiles and proficiency levels the K-level Ss bring to such a study. In consequence, he views assignment to treatment groups as a task for E, rather than as a task for an appropriately-instructed control and monitoring system. That is not to say that follow-on studies might not use the system to aid assignment of Ss to treatment groups.

Whether or not we give the system a role in assigning Ss to treatment groups, no useful purpose would be served by requiring the system to react to Cycle 1 entry skills data in an on-line manner. This is because assignment to treatment groups cannot occur until all Ss (N = 72) have been tested. If we treat S's time as valuable--which it is--then we will not have him sit around following entry skills testing while awaiting assignment to a treatment group. The obvious implication is that entry skills testing will occur on one day, with training initiating on a second day. Hence, assignment to groups can occur during an intervening 18-21 hour period. The most we could ask of the system if it were being used to control execution of the Koehler study is that it summarize Cycle 1 entry skills test data--by S, subtest, and code or code group--and to output these summaries while operating in an off-line mode following ET-1 testing of the last S.

---

will code the response either as acceptable or unacceptable--right or wrong. More complex evaluation schemes--e.g., that for the Koehler study--will distinguish between various levels or categories of unacceptability. The "real time" evaluated response typically is important in experiments only as a time saver; if E can evaluate the response quickly, then he might wish to record its evaluation code, rather than the response itself, thereby accomplishing one step in data reduction during conduct of the experiment. State of the art control-monitoring systems do not permit relieving E of this burden.

In the Koehler study, E will monitor S's response and will evaluate it when made. Where partial event control is accomplished by the system--as will be the case for IDCMS in Version 1 configuration--E will signal the evaluation code to the system. While this procedure has small implications for the data recording codes that the system will employ and for characteristics of the terminal through which E will address the system, there are no conditional implications. That is, required system reaction is the same whether E signals "Response is completed" or "Response is correct." Of passing interest, as executed, the Koehler study will both tape responses as made and record their evaluation codes immediately following completion of the response. The tape record will have a fail-safe function; it will be scrutinized only if anomalies show up in the reduced data.

Training Treatments

All of Koehler's training treatments stem from a knowledge-based, or phonics, orientation to beginning reading. Two primary training variables are studied. We denote these emphasis and strategy variables. Three categories of emphasis are distinguished: an analytic or segmentation emphasis (S), a synthetic or blending emphasis (B), and an analytic-synthetic or combined emphasis (C). (The mnemonic is SBC). Two categories of strategy are discerned: a single-letter strategy wherein segmentation responses go to the single-letter level (relatively) and blending responses come from that level--denoted 1--and a bipart strategy wherein segmentation responses go to a bipartite level and blending responses come from that level--denoted 2. Where the word is a VC item, the two strategies have identical implications. Where it is 3-letter or 4-letter, the single-letter strategy really is a tripartite strategy, which contrasts wth the bipartite strategy. Thus, the study deals (primarily) with a 3 x 2 matrix of training treatments:

	S1	B1	C1
	S2	B2	C2

Of passing interest, the study deals secondarily with a materials factor. Apparently-comparable but different materials--Versions 1 and 2--will be used. Each primary treatment group will be further subdivided into materials subgroups, having ns of 6--denoted S1.1, S1.2, . . . , C2.2. While it seems improbable that the system could handle a group of 6 Ss where one S belonged to each treatment group, we will ask it routinely to handle a group of 6 Ss half of whom are trained on Materials Set A and half on Materials Set B.

Four skills--T1 through T4--are addressed during training. T1 through T3 skills are taught to S groups, T3 and T4 to B groups, and T1 through T4 to C groups. These skills could be named as follows:

T1 - Pronouncing words as units

T2 - Segmenting words and pronouncing (sounding) the segments

T3 - Associating the elements of letter-sound rules

T4 - Combining and blending the sounds of segmented words

Table A-3 shows treatment groups by treatment materials sets. Noted earlier, each treatment group further subdivides into two content subgroups for materials. These subgroups receive comparable treatment materials but using different words reflecting different letter-sound rules.

Table A-3

Treatment Groups, by Training Materials

Skill	Training Materials	Treatment Groups
T1	T1.0	S1, S2, C1, C2
T2	T2.1	S1, C1
	T2.2	S2, C2
T3	T3.0	S1, S2, B1, B2, C1, C2
T4	T4.1	B1, C1
	T4.2	B2, C2

Following sections describe in order the characteristics of each training materials set, associated training times, and program inventories. But first we discuss some conventions.

Conventions Used to Establish Training Materials Requirements

1. Content-defined alternative treatment sets. Two versions of each set of treatment materials--Versions A and B--will be employed during every training session. Thus, if the session features T2 training using T2.1 materials, Version A of T2.1 will be used for training 3 Ss; Version B, for training 3 others.

2. Program item orders. The items of every program consisting of two or more items will occur in two alternative orders. Thus there will be two item-order referenced versions of each such program. Both of these versions will be employed during any session featuring the program to which these versions reference.

3. Audio programs. The system in present form cannot efficiently compose audio programs on-line from smaller elements. To do so requires quicker retrieval of audio elements from audio master reproduction than the system permits and the sequencing of these elements, whether prior to or during reproduction to audio buffer. An alternative is to store audio elements outside the system, to compose minimal nonredundant audio programs outside the system, and to load these programs into audio master reproduction. Single such programs permit retrieval and reproduction to audio buffer with delay on the order of 15 seconds on the average.

The Koehler study--and most studies involving instruction--features repeated trials for a minimal nonredundant program or--in some cases--repeated trials for such a program followed by a switched item order for the same program. Hence, we distinguish between single-trial programs--which are nonredundant for content--and double-trial programs--wherein the second trial portion repeats first trial content but with switched item order. Both types of program are taken as unitary entities for purposes of storage in audio master reproduction. Transferred to audio buffer, the single-trial program that is to be repeated over n trials will require use of a play, rewind, play, etc., sequence. The double-trial program will follow a similar sequence, except that a stop code intervening between the program's first and second trial portions will permit E to use whatever time he requires between trials. For present purposes, we assume that a program of either type in audio buffer will compel 5 seconds delay between trials.

4. General Instructions (A_0). It will be assumed that a general instruction A_0 will occur at the outset of any training program used during any cycle. Were we to tape A_0, then we would neither be able to take advantage of retrieval-reproduction delay nor be in a position to clarify any question that S might have. Hence, we assign the A_0 transmission function to E. An example of such an instruction keyed to T1 training is (to the effect) "In this task you are to look at the word on the slide and listen to how it is pronounced. Then you are to say the word." Maximum retrieval-reproduction delay will not exceed 30 seconds. That seems a reasonable time limit for A_0 and clarification of A_0. We assume that E will be able to prolong intertrial intervals for purposes of providing additional clarification if this is required, but will assume that the 5 second intertrial delay value will reflect the average such intertrial clarification requirement (audio buffer rewind delay is really less than 5 seconds for the short programs used in the Koehler study). Hence, training time calculations will reflect a general instructions time component that is 30 seconds plus 5(n - 1) seconds, with system switching delay viewed as concurrent to transmission of general instructions. (Some programs in addition will tape a very short instruction A_1 at the front of each item; this event is an integral part of item time, rather than of A_0 time.)

5. Housekeeping. Housekeeping consists of moving Ss into and out of the experimental situation and allied procedures not integral to the experiment as such. Housekeeping time will be computed for sessions, rather than for the running of programs per se.

6. Breaks. Break time will be computed for sessions, rather than for running of programs per se.

Since the Koehler study is used as the only basis for drawing implications for IDCMS when educational experiments are to be appreciably controlled by the system, one must ask whether these implications are simply ad hoc. One argument against the ad hoc characterization of the

present analysis is that although details of other educational experiments will differ from those for the Koehler study, other studies will stress the system in a comparable manner. Thus, for example, while intraitem event sequences may be more or less elaborate than those that Koehler employs, it may be that the only consequence is that item presentation-response time will increase or decrease accordingly. That is, intraitem event types and durations reflected in the Koehler study should characterize many SWRL studies. We know that this will not be universally true; for example, intraitem event durations must be much shorter in tachistoscopic recognition studies. Thus, while we may reject the view that the analysis will be ad hoc in the sense of having only the most-narrowly particular implications for IDCMS, we cannot in consequence accept the view that the analysis has implications of sufficient generality to encompass all contemplated educational experimentation at SWRL. One way to insure sufficiently general guidance on what IDCMS must be able to do is to augment the present guidance through analysis of other sorts of training--e.g., in music. Other papers might assess control implications while referencing to other sorts of training.

Set T1.0 Materials

T1 instruction will be given to the 48 Ss of Treatment Groups S1, S2, C1, and C2. All such instruction will use T1.0 materials. Ss will each receive 4 trials--on two items during Cycle 1, four items during Cycle 2, and three items during Cycle 3. We assume here that intertrial item order need not vary. Hence, the basic program will be an Item 1, . . . , Item n sequence that, in audio buffer, is presented four times in succession in consequence of play, rewind, play, etc., operations. A_0 time is $30 + 5(4 - 1)$, or 45, seconds.

Although item contents vary from one instructional program to the next and even from one item to the next, the sequence of events for any item of any trial of any training program of any cycle of the Koehler study tends to take the same form--although with recursion in the case of complex items. The item event sequence referencing to T1 instruction is shown in Table A-4.

The event sequence for a T1.0 item is graphed in Figure A-6. E first commands retrieval of T1.0 and duplication to audio buffer. Since this occurs during A_0 time, it need not be reflected in Figure A-6, which describes any one-item sequence for T1 instruction.

Consonant with foregoing assumptions, four versions of each T1.0 program are required--2 intersubject item orders x 2 content versions. The materials requirement is shown in Table A-5. Item and program lengths are in normal-play audio tape.

Table A-4

Event Sequence for T1 Instruction

Number	Descriptor	Event
1	Audio Instruction	A: "Repeat (say) the word after you hear it pronounced (spoken)."
2	Video + Audio Presentation	V: W A: /W/
3	Response	S_R: Try /W/
4	Audio Critique	A: /W/
5	"Evaluative" Feedback	E_F: E.g., "Stay with it."

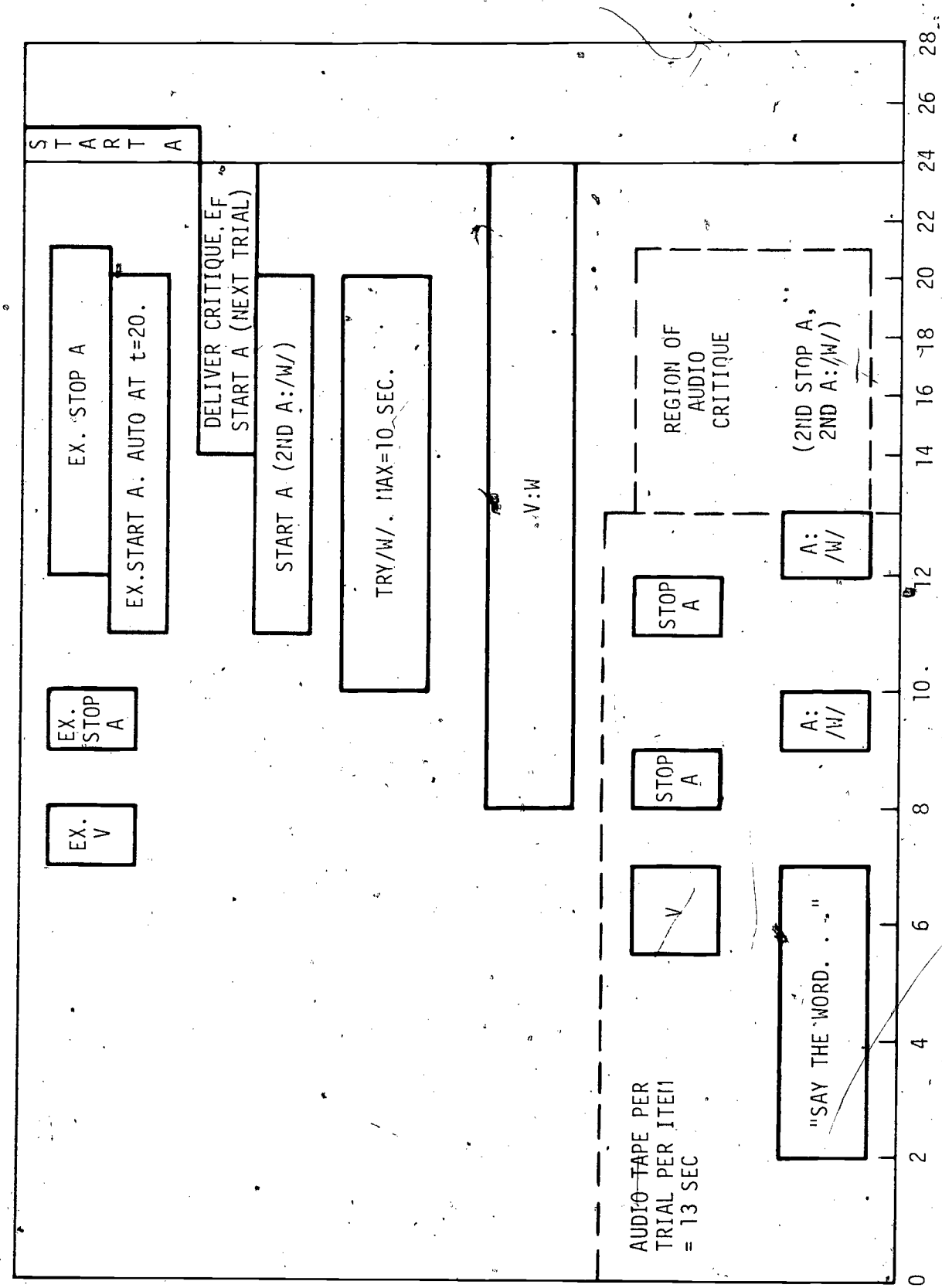


FIGURE A-6. SCHEMATIC REPRESENTATION OF EVENT CONTROL FOR T1.0 ITEMS.

Table A-5

T1.0 Materials Requirement

Cyc.	Tr.	Items	Length of Item (sec)	Length of Prog. (min)	Item Orders	Content Versions	Program Versions	V Frames per Vers.	Total V Frames
1	4	2	13	0.5	2	2	4	2	4
2	4	4	13	0.9	2	2	4	4	8
3	4	3	13	0.7	2	2	4	3	6

Although allowed 10 seconds to respond, S typically will respond in a very few seconds. We assume an average response time of 5 seconds--probably an inflated value. Hence, single-item, single-presentation time is on the order of 19 seconds (see Figure A-6). Table A-6 shows instructional time implications when the study is under system control and six Ss are instructed at a time.

Table A-6

T1.0 Instructional Time

Cycle	Tr.xIt.	Av. Negot. Time/Item (sec)	A_0 (sec)	Av. Negot. Time/Prog. (min)	No. Groups of 6	Predicted Tot. T1 Time (min)
1	8	19	45	3.3	8	26.4
2	16	19	45	5.8	8	46.4
3	12	19	45	4.6	8	36.8

Set T2.1 Materials

T2 instruction using T2.1 materials will be administered to S1 and C1 groups. T2.1 slides bear two printed stimuli--a word in normally-spaced form at the top of the slide and the same word in segmented form at the bottom of the slide--e.g.:

SAT
S-A-T

Two responses are required to the contents of such slides. The first is a whole-word pronunciation response to the top printed stimulus; the second, an appropriate sounding out of the word in segmented form. In both instances, S's response simply repeats an audio modelling of the response. To guide S concerning which printed stimulus is associated with the audio accompanyment that S is to repeat, the stimulus is boxed. Boxing of successive stimuli necessitates using two slides per item, rather than one. These two slides differ only in that the first boxes the top stimulus--e.g., SAT--while the second boxes the bottom stimulus--e.g., S-A-T.

The T2.1 item is a two-response item--as distinguished from the one-response items of T1 instruction and items to which we will come directly, which range up to six-response value. The Koehler study controls both for total training time per cycle per treatment group and for total number of presentation-response (or critiqued instructional response) entities. Hence, number of responses per item is relevant both to quantification of the materials requirement and of instructional time.

Event sequences for a presentation-response entity of T2 training using T2.1 materials have the same form as those for T1 training. The following sequence, which applies to the second, or segmented, stimulus of a Cycle 1 item, is illustrative.

A: "Repeat the (audio stimulus) after you hear it spoken."

V: C-V-C
A: /C/+/V/+/C/

S_R: Try /C/+/V/+/C/

A: /C/+/V/+/C/

E_F: E.g., "Simply amazing."

Timing of the event sequence should follow that for T1.0 items except that: a) per item costs will double for the two-response items of T2.1 and b) audio rendition of /C/+/V/+/C/ should cost two seconds rather than one. Hence, tape cost of the two-response item will be 30 seconds. The materials requirement is shown in Table A-7.

Consonant with greater complexity of the segmentation response, we assume T2.1 responses will average 6 seconds. Figure A-6 and foregoing comments support the view that average item negotiation time will be 2 x 22, or 44, seconds. Table A-8 shows instructional time implications when the study is under system control and six Ss are instructed at a time. A_0 computations are based on the 30 + 5(n - 1) seconds formula.

Set T2.2 Materials

T2 instruction using T2.2 materials will be administered to S2 and C2 groups. Materials differ from T2.1 materials primarily in how words are segmented, which is bipartitely, in number of stimuli per slide, and in item complexity. Table A-9 shows the pertinent data.

Consonant with conventions established above, tape cost will be 15 seconds per response--or 30 seconds per item for 2-response items and 45 seconds per item for 3-response items. The materials requirement is shown in Table A-10.

Table A-7

T2.1 Materials Requirement

Cyc.	Gr.	Tr.	It.	Rs	Item Leng. (sec)	Prog. Leng. (min)	Item Orders	Cont. Vers.	Prog. Vers.	V Frames per Vers.	Total V Frames
1	S1 C1	12 6	2	2	30	1.0	2	2	4	4	8
2	S1 C1	12 6	4	2	30	2.0	2	2	4	8	16
3	S1 C1	8 4	3	2	30	1.5	2	2	4	6	12

Table A-8

T2.1 Instructional Time

Cycle	Gr.	TxIxR	Av. Negot. Time/R (sec)	A_0 (sec)	Av. Negot. Time/Prog. (min)	No. Gr. of 6	Predicted Tot T2.1 Time (min)
1	S1	48	22	85	19.0	2	38.0
	C1	24	22	55	9.7	2	19.4
2	S1	96	22	85	36.6	2	73.2
	C1	48	22	55	18.5	2	37.0
3	S1	48	22	65	18.7	2	37.4
	C1	24	22	45	9.1	2	18.2

Table A-9

Programs, Response Levels, and Illustrative Items
for T2 Training Using T2.2 Materials

Cycle	Program	No. Rs	Illustration
1	1.1	3	SAT S-AT A-T
	1.2	2	SAT S-AT
2	2.1	2	SIN S-IN
	2.2	3	SPIN SP-IN I-N
	2.3	2	SPIN SP-IN
3	3.1	2	INK I-NK
	3.2	3	LINK L-INK I-NK
	3.3	2	LINK L-INK

Table A-10

T2.2 Materials Requirement

Cyc.	Prog.	Gr.	Tr.	It.	Rs	Item Leng. (sec)	Prog. Leng. (min)	Item Orders	Cont. Vers.	Prog. Vers.	V Frames per Vers.	Total V Frames
1	1.1	S2 C2	4 2	2	3	45	1.5	2	2	4	6	12
	1.2	S2 C2	6 3	2	2	30	1.0	2	2	4	4	8
2	2.1	S2 C2	11 5	2	2	30	1.0	2	2	4	4	8
	2.2	S2 C2	5 2	2	3	45	1.5	2	2	4	6	12
	2.3	S2 C2	6 3	2	2	30	1.0	2	2	4	4	8
3	3.1	S2 C2	7 3	2	2	30	1.0	2	2	4	4	8
	3.2	S2 C2	4 2	1	3	45	0.8	2	2	4	3	6
	3.3	S2 C2	4 2	1	2	30	0.5	2	2	4	2	4

Again we assume an average negotiation time per response of 22 seconds. Table A-11 shows instructional time implications when the study is under system control and six Ss are instructed at a time.

Set T3.0 Materials

All 72 Ss receive T3 training using T3.0 materials. All Ss receive this training in the same amount. The same 6 letter-sound combinations are taught during Cycles 1 and 2, with 2 additional letter-sound combinations added to the set during Cycle 3 training. During Cycles 1 and 2, the set of 6 rules is split into subsets of 3 rules each for preliminary training purposes. S first receives 2 trials on each rule of the first subset, but with randomization across trials. Next he receives 2 trials on each rule of the second subset under the same procedure. Finally, he receives 4 trials on each rule for the set as a whole. The same procedure is followed in Cycle 3 except that set size is 8 and subset size is 4. Again, two materials sets are used. Table A-12 illustrates the materials requirement for one materials set. Here it is important that a degree of trial-to-trial randomization of items occur. Half of the odd-numbered Ss might receive the odd-numbered programs of Table 4; half of the even-numbered Ss, the even-numbered programs. The remaining Ss would receive comparable programs from a second materials set. (Of passing interest, these materials lend themselves well to on-line composition of randomized sequences. However, if the system has no such capability, then one must preform the sequences and store these in audio master reproduction as required.) The last column of Table A-12 illustrates the rewind-replay operation for a program stored in audio buffer. That is, when four trials must occur, one rewinds the two-trial program and repeats the program. Stop codes should intervene between the single-trial components of these programs.

The same sequence of events referencing to presentation of a given letter-sound combination characterizes T3 training as characterized presentation of a given item or part-item during T1 and T2 training. Again video-audio presentation is followed by S's attempt to repeat audio. This in turn is followed by a critiquing representation of audio and evaluative feedback from E. Perhaps the only difference is that average response time should be shorter--let us say 3 seconds--and evaluative feedback shorter also--let us say 5 seconds. It also appears tenable that the audio instruction fronting each item should drop out. In consequence, audio tape per item should be on the order of 7.5 seconds. Allowing 5 seconds of tape for stop intervals, programs of the A1-A4 type will be 50 seconds long; those of the B1-B4 type, 65 seconds; those of the A5-A6 type, 95 seconds; those of the B5-B6 type, 125 seconds. The materials requirement is shown in Table A-13.

Consonant with foregoing remarks, presentation-response time per item should be on the order of 15 seconds. Table A-14 shows instructional time implications when the study is under system control and six Ss are instructed at a time.

Table A-11

T2.2 Instructional Time

Group	Cycle	Program	TxIxR	Av. Neg. Time/R (sec)	A_0 (sec)	Av. Neg. Time/Prog. (min)	No. Gr. of 6	Predicted Total T2.2 Time (min)
S2	1	1.1	24	22	45	9.6	2	19.2
		1.2	24	22	55	9.7	2	19.4
	2	2.1	44	22	80	17.5	2	35.0
		2.2	30	22	50	11.8	2	23.6
		2.3	24	22	55	9.7	2	19.4
	3	3.1	28	22	60	11.3	2	22.6
		3.2	12	22	45	5.2	2	10.4
		3.3	8	22	45	3.7	2	7.4
C2	1	1.1	12	22	35	5.0	2	10.0
		1.2	12	22	40	5.1	2	10.2
	2	2.1	20	22	50	8.2	2	16.4
		2.2	12	22	35	5.0	2	10.0
		2.3	12	22	40	5.1	2	10.2
	3	3.1	12	22	40	5.1	2	10.2
		3.2	6	22	35	2.8	2	5.6
		3.3	4	22	35	2.1	2	4.2

Table A-12

Two-Trial T3.0 Programs Featuring Intertrial Switching of Item Order

Cycle	Program[a]	Trials 1-2	Trials 3-4
1-2	A1	SPA-PAS	
	A2	APS-PSA	
	A3	NTI-TIN	
	A4	ITN-TNI	
	A5	SANTIP-TNAPSI	Repeat A5
	A6	APNITS-PASTIN	Repeat A6
3	B1	KSAP-SKPA	
	B2	AKPS-PASK	
	B3	NLIT-LNTI	
	B4	TILN-ILTN	
	B5	LSATKNIP-KSLINTPA	Repeat B5
	B6	STNPLAKI-PNLKATIS	Repeat B6

[a]All of these programs belong to one content version. The other version uses different consonant letter-sound rules. Those who receive the Table A-12 version receive only half of the tabled programs--the odd-numbered ones or the even-numbered ones. These programs differ from previous ones in being of two-trial length, to accommodate a requirement for intertrial switching of item order without incurring the usual retrieval-reproduction delay between trials.

Table A-13

T3.0 Materials Requirement

Cycle	Program	Trials	Rs/Trial	Prog. Leng. (min)	Item Orders	Cont. Vers.	Prog. Vers.	V Frames /Version	Total V Frames[a]
1-2	1.1	2	3	.8	2	2	4		
	1.2	2	3	.8	2	2	4		
	1.3	4	6	1.6	2	2	4	6	12
3	2.1	2	4	1.1	2	2	4		
	2.2	2	4	1.1	2	2	4		
	2.3	4	8	2.1	2	2	4	2+ Prog. 1.3	4+ Prog. 1.3

[a]If there were not content overlap between content versions, then each version would require 8 video frames and the two together 16. If, as is likely, the two overlap for vowel letter-sound rules, then the unique total V frame requirement is 14.

Table A-14

T3.0 Instructional Time

Cycle	Program	TxR	Av. Neg. Time/R (sec)	A_0 (sec)	Av. Neg. Time/Prog. (min)	No. Gr. of 6	Predicted Tot. T3.0 Time
1	1.1	6	15	35	2.1	12	25.2
	1.2	6	15	35	2.1	12	25.2
	1.3	24	15	45	6.8	12	81.6
2	1.1				2.1		25.2
	1.2				2.1		25.2
	1.3				6.8		81.6
3	2.1	8	15	35	2.6	12	31.2
	2.2	8	15	35	2.6	12	31.2
	2.3	32	15	45	8.8	12	105.6

Set T4.1 Materials

T4 instruction using T4.1 materials is given to Groups B1 and C1. Table A-15 shows the forms of items across cycles.

Table A-15

Illustrative T4.1 Items

Cycle	Illustrative Item
1	P-A-N PAN
2	SP-A-N SPAN
3	S-A-NK SANK

The basic event sequence is as indicated for earlier instruction. Setting tape per response at 15 seconds, then the two-response items of T4 training using T4.1 materials use 30 seconds of tape. The materials requirement is shown in Table A-16.

Assuming as earlier an average negotiation time per response of 22 seconds, Table A-17 shows instructional time implications when the study is under system control and six Ss are instructed at a time.

Set T4.2 Materials

T4 instruction using T4.2 materials is given to Groups B2 and C2. Although segmentation is bipartite rather than tripartite, materials are more extensive than for training using T4.1 materials. The materials for one content version of T4.2 programs is shown in Table A-18.

A T4.2 program of the Program 1.1 type requires S to respond 4 times to each of 2 items. Items range from 2-response (e.g., Program 1.2) to 6-response (Program 2.2). The T4.2 materials requirement is shown in Table A-19. Again we assume 15 seconds of tape per response.

Assuming again an average negotiation time per response of 22 seconds, Table A-20 shows instructional time implications when the study is under system control and six Ss are instructed at a time.

Table A-16

T4.1 Materials Requirement

Cyc.	Gr.	Tr.	It.	Rs	Item Leng. (sec)	Prog. Leng. (min)	Item Orders	Cont. Vers.	Prog. Vers.	V Frames /Prog.	Total V Frames
1	B1 C1	14 6	2	2	30	1.0	2	2	4	4	8
2	B1 C1	14 6	4	2	30	2.0	2	2	4	8	16
3	B1 C1	15 6	2	2	30	1.0	2	2	4	4	8

Table A-17

T4.1 Instructional Time

Cycle	Group	TxIxR	Av. Neg. Time/R (sec)	A_0 (sec)	Av. Neg. Time/Prog. (min)	No. Gr. of 6	Predicted Tot. T4.1 Time
1	B1	56	22	95	22.1	2	44.2
	C1	24	22	55	9.7	2	19.4
2	B1	112	22	95	42.7	2	85.4
	C1	48	22	55	18.5	2	37.0
3	B1	60	22	100	23.7	2	47.4
	C1	24	22	55	9.7	2	19.4

Table A-18

T4.2 Materials for One Content Version

Cycle	Program	Items for One Content Version[a] (1)	(2)	(3)	(4)
1	1.1	A-N	I-T		
		AN	IT		
		P-AN	S-IT		
		PAN	SIT		
	1.2	P-AN	S-IT		
		PAN	SIT		
2	2.1	A-N	I-P		
		AN	IP		
		SP-AN	T-IP		
		SPAN	TIP		
	2.2	A-P			
		AP			
		N-AP			
		NAP			
		SN-AP			
		SNAP			
	2.3	SP-AN	T-IP	N-AP	SN-AP
		SPAN	TIP	NAP	SNAP
3	3.1	A-NK	I-NT		
		ANK	INT		
		S-ANK	T-INT		
		SANK	TINT		
	3.2	S-ANK	T-INT		
		SANK	TINT		

[a]Single items read down. Responses per item range from 2 to 6. The content version employs 14 unique printed stimulus layouts. Since these must appear with one or the other of the two printed stimuli boxed, the content version implies 28 video frames. Two content versions would require 56 video frames if the second version showed the same overlapping pattern across programs as does the first.

Table A-19

T4.2 Materials Requirement

Cycle	Program	Gr.	Tr.	It.	Rs	Item Leng. (sec)	Prog. Leng. (min)	Item Orders	Cont. Vers.[a]	Prog. Vers.[b]
1	1.1	B2 C2	4 2	2	4	60	2.0	2	2	4
	1.2	B2 C2	6 2	2	2	30	1.0	2	2	4
2	2.1	B2 C2	4 2	2	4	60	2.0	2	2	4
	2.2	B2 C2	5 2	1	6	90	1.5	1	2	2
	2.3	B2 C2	6 3	4	2	30	2.0	2	2	4
3	3.1	B2 C2	4 2	2	4	60	2.0	2	2	4
	3.2	B2 C2	7 3	2	2	30	1.0	2	2	4

[a]Since a Program 2.2 contains only one item, there can only be one intersubject item order for it. Hence, there are only two program versions for this type of program.

[b]Table A-18 reveals a requirement for 28 video frames per version and 56 overall. While Table A-18 can be used to show V frames per version for each of the seven programs per version of T4.2 materials, these values reflect a good deal of program-to-program overlap and are not presented here for that reason.

Table A-20

T4.2 Instructional Time

Cycle	Program	Groups	TxIxR	Av. Neg. Time/R (sec)	A_0 (sec)	Av. Neg. Time/Prog. (min)	No. Gr. of 6	Predicted Tot. T4.2 Time (min)
1	1.1	B2	32	22	45	12.5	2	25.0
		C2	16	22	35	6.5	2	13.0
	1.2	B2	24	22	55	9.7	2	19.4
		C2	8	22	35	3.5	2	7.0
2	2.1	B2	32	22	45	12.5	2	25.0
		C2	16	22	35	6.5	2	13.0
	2.2	B2	30	22	50	11.8	2	23.6
		C2	12	22	35	5.0	2	10.0
	2.3	B2	48	22	55	18.5	2	37.0
		C2	24	22	40	9.5	2	19.0
3	3.1	B2	32	22	45	12.5	2	25.0
		C2	16	22	35	6.5	2	13.0
	3.2	B2	28	22	60	11.3	2	22.6
		C2	12	22	40	5.1	2	10.2

Magnitude of Training

Table A-21 shows total number of training responses and total training time, by treatment group and cycle. Response values coincide with those of the Koehler formulation. Training time values are those generated above. While the study in manual mode execution might not allot the values reached above for presentation-response sequences, study scheduling suggests that overall training time values are approximately those reached in Table A-21 and Table A-22. The study allocates six sessions to the training of any S. If each session uses 20-24 minutes for actual training, then study training time will correspond to that shown in Table A-22, which summarizes training across skills and cycles. Excepting for slightly longer S2 and C2 Cycle 2 and Cycle 3 training times, the tables show marked intracycle matches for training time across treatment groups.

Table A-21

Response and Training Time Totals, by Treatment Group and Cycle

Gr.	Materials	Cycle 1 Prog.	Cycle 1 Rs	Cycle 1 Tng. Time (min)	Cycle 2 Prog.	Cycle 2 Rs	Cycle 2 Tng. Time (min)	Cycle 3 Prog.	Cycle 3 Rs	Cycle 3 Tng. Time (min)
S1	T1.0	1.1	8	3.3	2.1	16	5.8	3.1	12	4.6
	T2.1	1.1	48	19.0	2.1	96	36.6	3.1	48	18.7
	T3.0	1.1	6	2.1	1.1	6	2.1	2.1	8	2.6
		1.2	6	2.1	1.2	6	2.1	2.2	8	2.6
		1.3	24	6.8	1.3	24	6.8	2.3	32	8.8
	Totals		92	33.3		148	53.4		108	37.3
B1	T3.0	1.1	6	2.1	1.1	6	2.1	2.1	8	2.6
		1.2	6	2.1	1.2	6	2.1	2.2	8	2.6
		1.3	24	6.8	1.3	24	6.8	2.3	32	8.8
	T4.1	1.1	56	22.1	2.1	112	42.7	3.1	60	23.7
	Totals		92	33.1		148	53.7		108	37.7
C1	T1.0	1.1	8	3.3	2.1	16	5.8	3.1	12	4.6
	T2.1	1.1	24	9.7	2.1	48	18.5	3.1	24	9.1
	T3.0	1.1	6	2.1	1.1	6	2.1	2.1	8	2.6
		1.2	6	2.1	1.2	6	2.1	2.2	8	2.6
		1.3	24	6.8	1.3	24	6.8	2.3	32	8.8
	T4.1	1.1	24	9.7	2.1	48	18.5	3.1	24	9.7
	Totals		92	33.7		148	53.8		108	37.4
S2	T1.0	1.1	8	3.3	2.1	16	5.8	3.1	12	4.6
	T2.2	1.1	24	9.6	2.1	44	17.5	3.1	28	11.3
		1.2	24	9.7	2.2	30	11.8	3.2	12	5.2
					2.3	24	9.7	3.3	8	3.7
	T3.0	1.1	6	2.1	1.1	6	2.1	2.1	8	2.6
		1.2	6	2.1	1.2	6	2.1	2.2	8	2.6
		1.3	24	6.8	1.3	24	6.8	2.3	32	8.8
	Totals		92	33.6		150	55.8		108	38.8

Table A-21 - continued

Gr	Materials	Cycle 1 Prog.	Cycle 1 Rs	Cycle 1 Tng. Time (min)	Cycle 2 Prog.	Cycle 2 Rs	Cycle 2 Tng. Time (min)	Cycle 3 Prog.	Cycle 3 Rs	Cycle 3 Tng. Time (min)
B2	T3.0	1.1	6	2.1	1.1	6	2.1	2.1	8	2.6
		1.2	6	2.1	1.2	6	2.1	2.2	8	2.6
		1.3	24	6.8	1.3	24	6.8	2.3	32	8.8
	T4.2	1.1	32	12.5	2.1	32	12.5	3.1	32	12.5
		1.2	24	9.7	2.2	30	11.8	3.2	28	11.3
					2.3	48	18.5			
	Totals		92	33.2		146	53.8		108	37.8
C2	T1.0	1.1	8	3.3	2.1	16	5.8	3.1	12	4.6
	T2.2	1.1	12	5.0	2.1	20	8.2	3.1	12	5.1
		1.2	12	5.1	2.2	12	5.0	3.2	6	2.8
					2.3	12	5.1	3.3	4	2.1
	T3.0	1.1	6	2.1	1.1	6	2.1	2.1	8	2.6
		1.2	6	2.1	1.2	6	2.1	2.2	8	2.6
		1.3	24	6.8	1.3	24	6.8	2.3	32	8.8
	T4.2	1.1	16	6.5	2.1	16	6.5	3.1	16	6.5
		1.2	8	3.5	2.2	12	5.0	3.2	12	5.1
					2.3	24	9.5			
	Totals		92	34.4		148	56.1		110	40.2

Table A-22

Summary of Response and Training Time Totals Across Skills and Cycles

Gr.	Cyc.	T1.0		T2.1		T2.2		T3.0		T4.1		T4.2		Totals	
		Rs	Time	Rs	Time	Rs	Time	Rs	Time	Rs	Time	Rs	Time	Rs	Time
S1	1	8	3.3	48	19.0			36	11.0					92	33.3
	2	16	5.8	96	36.6			36	11.0					148	53.4
	3	12	4.6	48	18.7			48	14.0					108	37.3
Totals		36	13.7	192	74.3			120	36.0					348	124.0
B1	1							36	11.0	56	22.1			92	33.1
	2							36	11.0	112	42.7			148	53.7
	3							48	14.0	60	23.7			108	37.7
Totals								120	36.0	228	88.5			348	124.5
C1	1	8	3.3	24	9.7			36	11.0	24	9.7			92	33.7
	2	16	5.8	48	18.5			36	11.0	48	18.5			148	53.8
	3	12	4.6	24	9.1			48	14.0	24	9.7			108	37.4
Totals		36	13.7	96	37.3			120	36.0	96	37.9			348	124.9

Table A-22 - continued

Gr.	Cyc.	T1.0 Rs	T1.0 Time	T2.1 Rs	T2.1 Time	T2.2 Rs	T2.2 Time	T3.0 Rs	T3.0 Time	T4.1 Rs	T4.1 Time	T4.2 Rs	T4.2 Time	Totals Rs	Totals Time
S2	1	8	3.3			48	19.3	36	11.0					92	33.6
	2	16	5.8			98	39.0	36	11.0					150	55.8
	3	12	4.6			48	20.2	48	14.0					108	38.8
Totals		36	13.7			194	78.5	120	36.0					350	128.2
B2	1							36	11.0			56	22.2	92	33.2
	2							36	11.0			110	42.8	146	53.8
	3							48	14.0			60	23.8	108	37.8
Totals								120	36.0			226	88.8	346	124.8
C2	1	8	3.3			24	10.1	36	11.0			24	10.0	92	34.4
	2	16	5.8			44	18.3	36	11.0			52	21.0	148	56.1
	3	12	4.6			22	10.0	48	14.0			28	11.6	110	40.2
Totals		36	13.7			90	38.4	120	36.0			104	42.6	350	130.7

NATURAL LANGUAGE ANALYSIS

TM 5-71-02. Ann Porch. An Analysis of Methods for Preparing a Large Natural Language Data Base, February 16, 1971.

Relative cost and effectiveness of techniques for preparing a computer-compatible data base consisting of approximately one million words of natural language are outlined. Considered are dollar cost, ease of editing, and time consumption. Facility for insertion of identifying information within the text, and updating of a text by merging with another text are given special attention. It is concluded that MTST and Telterm2 are two highly effective methods of text preparation. The decision of which to use on a particular project would depend on available funds and possible peripheral uses for the equipment. Criteria for making such a decision are discussed.

TN 5-71-39. Lanaii Kline. QUES (Question Survey), June 11, 1971.

QUES is a computer program which answers predetermined questions about an information bank of an IMS reading program. There are eight questions, each represented by a subroutine to the main program. Part of the variable input controls the program flow, i.e., controls which questions are to be queried and the number of times variable input will be provided for each question. Thus, the user can specify that question #1 be queried with two sets of variable input, question #4 be queried with one set of variable input, etc.

TN 2-71-10. Carol Pfaff. Naturalistic Observational Dialect Studies: Processing and Output Format Requirements, July 21, 1971.

This paper is part of a series documenting data processing requirements for naturalistic observational dialect studies. Companion documents are Russell (1970) and Legum and Pfaff (1971). This paper describes major processing routines involved in the formation of KWICs, CODE TABLES, and FREQUENCY LISTS, and specifies the desired output format for each. Aspects of input format which are relevant to programming these processing types are also specified.

TN 2-71-11. Stanley E. Legum and Carol Pfaff. Naturalistic Observational Dialect Studies: Additional Requirements and Usage Estimates, July 29, 1971.

Part I of this paper describes automated "housekeeping" requirements associated with large scale naturalistic observational dialect studies. Part II describes the special requirements necessitated by unequal sample sizes. Part III estimates the relative frequency of usage of each of the major routines which are required as a function of the number of groups and speakers studied. Russell (1970) and Pfaff (1971) are required reading to place this document in perspective.

TM 5-71-10. Ann Porch. People and Projects in Natural Language Processing: A Preliminary Bibliographic Directory, August 5, 1971.

This document is part of a continuing survey of the field of natural language processing. It presents an overview of applications in education, literature, computational linguistics, information retrieval, machine translation, and artificial intelligence. Primary consideration is given to researchers working or publishing in the United States since 1960, with the bulk of the citations confined to the past three years. Addresses and phone numbers are supplied where such information is available. Source materials are largely secondary, with most references derived from specialized periodical bibliographies in the field. The directory is prepared in computer-compatible form for ease of updating.

TN 5-71-91. Ann Porch. Comparative Code Chart (Interim Version), August 10, 1971.

A chart is given which allows the user to make rapid and easy comparisons among the following commonly used computer-oriented forms of coding date: binary, octal, decimal, hexidecimal, ASCII, BCD, EBCDIC, paper tape, punch cards (026 and 029). A brief discussion is given of the manner in which the chart is intended to be utilized and of conventions used in its compilation.

TN 5-71-56. Lanaii Kline. Essay Word Count and Statistics Program, August 18, 1971.

This program searches an essay for unique words and maintains a frequency count for each unique word. Three reports are prepared:

(1) a frequency ordered list
(2) an alphabetized word list
(3) a word/sentence statistical breakdown

This program is an expanded version of the ESSAY WORD COUNT PROGRAM (TN 5-70-12).

TN 5-71-74. Ann Porch. Preliminary Design for a Language Analysis Package (L.A.P.), August 18, 1971.

A package of computer programs to handle natural language retrieval analysis in a manner analogous to that of a statistical package is discussed. The document presents an overview of several language analysis projects currently underway, and of several research approaches to resolving problems in language analysis. Emphasis is presented on the way the L.A.P. could be used with each approach. Design specification for a package sensitive to the state-of-the-art are documented. A preliminary implementation, using programs in SWRL's possession, is suggested as a first step in the development process. An overview of the design considerations and algorithms for the preliminary package is presented.

TN 5-72-01. Lanaii Kline. Story-List, January 21, 1972.

This document describes the program, STORY-LIST which prepares an alphabetized cumulative vocabulary list for each story within a school grade.

TN 5-72-10. Lanaii Kline. CONC-Dictionary, February 29, 1972.

The program sets up three dictionaries; basic, fiction, and fantasy words. It then processes each essay by parsing the essay into words, checking if the word is a legal word (in the dictionary), and adding it to the essay unique-word list.

TN 5-72-27. Ann Porch. Module Design Document: Scan Module - L.A.P. Version I, April 24, 1972.

This is one of a series of technical design specifications for individual modules in the Language Analysis Package (L.A.P.).

The Scan Module will read input text, divide it into words, and check for special characters indicating further action needed.

PERIPHERAL SYSTEMS

Data Entry

TN 5-71-29. Ann Porch. A Hardware Configuration for a Flexible, Multi-Purpose Data Entry Station, May 13, 1971.

The SWRL Computer Center recently acquired hardware for a data-entry station of high flexibility and wide-spread applicability. The system employs a Cathode Ray Tube (CRT) video terminal with keyboard combined with two incremental cassette recorders, a standard teletypewriter, a modem for computer communication, and a switching connection box for controlling the flow of data. It can be used either for entering data on-line to a computer, or off-line as an editing station for preparing computer compatible materials.

This document is intended to serve as a system description and an operating manual. It does not attempt to replace the manuals associated with the individual components of the data entry station, but rather to coordinate and condense the information contained within them into a single presentation of the system as a whole. The presentation is divided into two major parts: (1) a description of the system; (2) a manual for operations.

TN 5-71-86. Ashok Dave. Analysis of Presently Available Optical Mark Readers, August 18, 1971.

This document analyzes presently available Optical Mark Readers on the basis of evaluation factors discussed in TN 5-71-90.

TN 5-71-90. Ashok Dave. Factors Relevant to the Evaluation of IMS-Data-Input-Compatible Optical Mark Readers, August 18, 1971.

Several types of optical readers using various techniques and having wide applications are presently available on the market. An attempt has been made to establish criteria and information which will assist in selecting compatible equipment that is cost-effective with IMS.

TM 5-71-11. Ashok Dave. Considerations in Selecting Keyboards for Man-Machine Interaction, November 3, 1971.

Factors discussed include keyboard layout, character set, accessibility, key characteristics, and control/function keys.

Remote Job Entry

TN 5-71-42. Sheridan Bentson. User's Guide for Remote Job Entry to Jet Propulsion Laboratory, June 28, 1971.

The Remote Job Entry (Interim Version) into Jet Propulsion Laboratory's UNIVAC 1108 Computer is operating on SWRL's Command 690 System. This document explains the use and restrictions of the RJE as it presently exists.

TN 5-71-43. Sheridan Bentson. Status Report on 690/1108 Remote Job Entry Computer Software (Interim Version), June 28, 1971.

The Interim Version of the Remote Job Entry to Jet Propulsion Laboratory's UNIVAC 1108 Computer is operational on SWRL's Command 690 System. This report is a Programmer's Guide to the present RJE system, indicating the status of the present computer software.

TN 5-71-79. Sheridan Bentson. Calculating Theoretical Transmission Speeds of a Remote Job Entry, August 6, 1971.

An upper bound on the data transfer rate in a computer remote job entry system can be calculated by considering transmission in an error-free environment. This Technical Note presents a function for making such a computation.

TN 5-72-06. Sheridan Bentson. Programmer's Guide to the Remote Job Service to UCLA, January 31, 1972.

This report contains a technical description of the 690 software system 'RJS', which offers remote job entry terminal service from SWRL's 690 to UCLA's IBM 360/91.

TM 5-72-07. Sheridan Bentson. Terminal Job Entry, July 6, 1972.

SWRL's Terminal Job Entry (TJE) is a computer software system for the Command 690. TJE allows medium-speed operation of a timesharing Central Computer, using the Command 690's peripheral devices.

STUDENT-SYSTEM INTERACTION

TN 5-71-23. Milton Schwartz. Display Terminal Oriented Software (DTOS) Interim User's Manual, April 27, 1971.

DTOS is a generalized library subroutine package designed to facilitate Telterm II CRT display manipulation from a computer. The routines are written in EXFOR (International Timesharing Corporation's Extended Fortran IV). EXFOR contains many advanced features not available in standard Fortran IV.

TN 5-71-30. Milton Schwartz. Hardware Specifications and Operation of the Random Access Audio Device, May 19, 1971.

This TN reports on the hardware aspects of the Random Access Audio Device (RAAD) being evaluated for use as a component in an experimental Instructional Learning Station under development at the SWRL Computer Center. The RAAD has been interfaced with a teletype terminal which operates on a dial-up mode to a remote time-shared computer. The computer has been programmed to selectively call out prestored audio messages from the RAAD to effectively simulate a learning situation and present reports on learning performance.

TN 5-71-41. Ashok Dave and Frank Teplitzky. Exploratory Work With the Brobeck and Associates Random-Access Audio Device, June 23, 1971.

This document describes the functional aspects of the Brobeck and Associates Random-Access Audio Device, the computer software requirements to access it, and the scripts prepared in connection with developmental tests of the hardware/software functions.

TN 5-71-45. Ashok Dave and Frank Teplitzky. Overview of Audio Response Systems, July 16, 1971.

This document presents an overview of computer-controllable audio response systems based on information extracted from vendor supplied brochures and recent literature.

TN 5-71-75. Ashok Dave and Milton Schwartz. Overview of Visual Display Systems, August 18, 1971.

An overview of computer-controllable visual display systems based on information extracted from vendor supplied brochures and recent literature.

TN 5-71-88. Ashok Dave. Cost-Storage Capacity Analysis of Mini-Computer Compatible Magnetic Disks, August 18, 1971.

This document describes the characteristics of various kinds of magnetic disk storage devices, and presents a cost-storage capacity analysis of mini-computer compatible disks.

TN 1-72-01. Joseph F. Follettie. Experimental and Student-System Interactive Instructional Illustrations Pertinent to IDCMS, February 4, 1972.

SWRL IDCMS will be a flexible hardware system for controlling and monitoring instruction and research in the Laboratory setting. This paper seeks to introduce potential users to the system and software designers to representative challenges that system exploitation will pose.